Package 'RSAtools'

May 7, 2026

Type Package

Title Advanced Response Surface Analysis

Version 0.1.2

Author Fernando Núñez-Regueiro [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4784-2021>),
Jacques Juhel [aut] (ORCID: <https://orcid.org/0000-0002-3520-6012>),
Felix Schönbrodt [ctb] (ORCID: <https://orcid.org/0000-0002-8282-3910>),
Sarah Humberg [ctb] (ORCID: <https://orcid.org/0000-0002-7891-3622>)

Maintainer Fernando Núñez-Regueiro <fernando.nr.france@gmail.com>

Description
Provides tools for response surface analysis, using a comparative framework that identifies best-fitting solutions across 37 families of polynomials. Many of these tools are based upon and extend the 'RSA' package, by testing a larger scope of polynomials (+27 families), more diverse response surface probing techniques (+acceleration points), more plots (+line of congruence, +line of incongruence, both with extrema), and other useful functions for exporting results.

Suggests fields, rgl, qgraph, tcltk, tkrplot, testthat, covr, psych, ggpubr

Depends R (>= 2.15.0), lavaan (>= 0.5.20), semTools (>= 0.5-5), RSA (>= 0.10.4), ggplot2, lattice

Imports plyr, RColorBrewer, aplpack, methods, MASS

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2025-09-01 20:30:02 UTC

Contents

`best.rsa` *Compare a list of polynomial models against the data*

Description

Compares any number of predefined or user-specific polynomial models and extracts their fit indices, thereby establishing best-fitting solutions.

Usage

```
best.rsa(RSA_object, order = c("wAIC", "R2adj"), robust = F)
```

Arguments

`RSA_object`	x an object of class "RSA_object" generated by RSAmodel(). If x is already generated by best.rsa(), then best.rsa will simply apply reordering according to "order".
`order`	Single or vector of fit indices used to determine best-fitting polynomial families. The output matrix is ordered based on this fit index
`robust`	Should robust fit indices should be extracted? (default= TRUE)

Details

This function compares models based on information-theoretic criteria and statistical tests. The cubic saturated polynomial provides a benchmark reference for fit, against which predefined polynomial families (37 to date) or user-specific variants of these families are compared for absolute fit (likelihood ratio test), parsimony (wAIC), explained variance (adjusted R2), and ordinary SEM criteria (e.g., CFI, TLI, RMSEA, SRMR).

Value

A table containing fit indices for each model

Examples

```
#####ESTIMATE RSA OBJECT
RSA_step1 <-  RSAmodel(engagement ~ needs*supplies,
data= sim_NSfit, model= c("CUBIC","FM8_INCONG","FM9_INCONG","FM20_ASYMCONG",
"FM21_ASYMCONG","FM26_PARALLELASYMWEAK"))
##### COMPARE POLYNOMIAL FAMILIES FROM THE RSA OBJECT
RSA_step1_fit <- best.rsa(RSA_step1,order=c("wAIC"))
names(RSA_step1$models)
#Inspect best-fitting family model
summary(RSA_step1$models$FM26_PARALLELASYMWEAK)
```

`best.rsa2` *Compare two polynomial models against the data*

Description

Compare two polynomial models, for example to test parametric constraints in STEP2 of the 3-step identification strategy (see `RSAmodel`).

Usage

```
best.rsa2(RSA_object, m1, m2, order = c("wAIC"), robust = F)
```

Arguments

`RSA_object`	x an object of class "RSA_object" generated by RSAmodel()
`m1`	First model to be compared (contained in RSA_object$models)
`m2`	Second model to be compared (contained in RSA_object$models)
`order`	Fit index used to determine best-fitting model.
`robust`	A boolean stating whether robust fit indices should be extracted (default= TRUE)

Value

A table containing fit indices for each model

Examples

```
##### Test a variant within a family (e.g., FM26_PARALLELASYMWEAK)
##Define variant as constraints
list_variant <- list()
list_variant[["variant1"]] <- c('
####First-order polynomials: variant-specific
b1 == -1/4*b2
####Second-order polynomials: variant-specific
b5 == b3/2
b4 == 0
####Third-order polynomials: FM26_PARALLELASYMWEAK
```

```
b6 == 0
b7 == 0
b9 == b8/-3
')
RSA_NSfit  <- RSAmodel(formula= engagement ~ needs*supplies,
data= sim_NSfit, model= c("FM26_PARALLELASYMWEAK","USER"),
user_model= list_variant)
##Compare variant to best-fitting family (e.g., LRT_pvalue p > .05)
best.rsa2(RSA_NSfit,m1="variant1",m2="FM26_PARALLELASYMWEAK")[2,1:3]
```

`exportRSA.bootstrap` *Export bootstrapped parameters of polynomial model*

Description

Utilitary function to export bootstrapped parameters, as part of STEP3 of the 3-step identification strategy (see `RSAmodel`). Utilitary function to export bootstrapped parameters

Usage

```
exportRSA.bootstrap(RSAbootstrap_object)
```

Arguments

`RSAbootstrap_object`
: A matrix output generated by lavaan: `bootstrapLavaan(RSA_object$models$name_final, FUN="coef")`

Value

A table of mean values and 95

Examples

```
###### Export 95% CI of bootstrapped estimates of polynomial
#Estimate a model: FM26_PARALLELASYMWEAK (simulation data)
RSA_NSfit  <- RSAmodel(formula= engagement ~ needs*supplies,
data= sim_NSfit, model= c("FM26_PARALLELASYMWEAK"))
#Bootstrapped sampling with lavaan
RSA_NSfit_boot  <- lavaan::bootstrapLavaan(RSA_NSfit$models$FM26_PARALLELASYMWEAK,
 R= 10,FUN="coef")
#Export results in a table
RSA_NSfit_boot_exp <- exportRSA.bootstrap(RSA_NSfit_boot)
RSA_NSfit_boot_exp
```

exportRSA.fit *Export fit indices and names of polynomials models*

Description

Utilitary function to easily export fit indices of polynomial models (predefined or user-specific), including their proper theoretical names (for predefined polynomial families) and the fit indices used for comparison.

Usage

```
exportRSA.fit(best.rsa_object)
```

Arguments

`best.rsa_object`
: A list generated by `best.rsa(RSA_object)`

Value

A table of fit indices and names of polynomial families or user model

ident.ext *Probe extrema in the response surface*

Description

Identify reversal or acceleration points (generically called "extrema") in the LOC or LOIC of the response surface and test how many of them have outcome observations that significantly differ from what would be expected for predictor combinations on these points (that have the same level)

Usage

```
ident.ext(
  RSA_object,
  model = NULL,
  acceleration = c(0, 0),
  alpha = 0.05,
  z_tested = "observed",
  alphacorrection = "none",
  n_sample = NULL,
  verbose = TRUE,
  df_out = FALSE
)
```

Arguments

RSA_object	An RSA object
model	The model to be probed for extrema (reversal or acceleration points)
acceleration	Rates of accelerations along the LOC and LOIC to be inspected (0< abs(rate) < 1). Acceleration points will only appear if reversals do not exist, and if acceleration rates exist (if not, a warning will appear).
alpha	Alpha level for the one-sided confidence interval of the outcome predictions on the extrema
z_tested	Should significance tests be conducted on "observed" or "predicted" observation
alphacorrection	Set "Bonferroni" to adjust the alpha level for multiple testing when testing the outcome predictions of all data points behind the extrema
n_sample	Number of random draws to consider to find extrema. This option is used for large samples to increase speed in preliminary analyses, but it is not recommended for published results). Defaults to NULL.
verbose	Should extra information be printed?
df_out	Number of random draws to consider to find extrema. This option is used for large samples to increase speed in preliminary analyses, but it is not recommended for published results). Defaults to NULL.

Details

When testing for reversals or accelerations in nonlinear response surfaces involving quadratic or cubic polynomial families (FM4 to FM37), the RSAextrema function helps to determine the exact location of reversal or acceleration points along the lines of congruenc (LOC) or incongruence (LOIC), and the number and percentage of observations significantly affected by these reversals or accelerations (for a given probability level, alpha). This points are determined according to derivatives of the function according to rationales for combining polynomials (Núñez-Regueiro & Juhel, 2022, 2024).

Value

A table containing the location and percentages of observations above or below extrema

References

Núñez-Regueiro, F., Juhel, J. (2022). *Model-Building Strategies in Response Surface Analysis* Manuscript submitted for publication.

Núñez-Regueiro, F., Juhel, J. (2024). *Response Surface Analysis for the Social Sciences II: Combinatory Rationales for Complex Polynomial Models* Manuscript submitted for publication.

See Also

plotting.ext, RSAmodel

plotting.ext *Plot extrema in the response surface along the lines of congruence and incongruence*

Description

Plots the response surface of an RSA object along the lines of congruence (LOC) and incongruence (LOIC), while indicating the location of reversal or acceleration points in the surface (when they exist).

Usage

```
plotting.ext(
  RSA_object,
  model,
  acceleration = c(0, 0),
  n_sample = 100,
  names_xLOC = NULL,
  names_xLOIC = NULL,
  names_z = NULL,
  xlim = NULL,
  zlim = NULL,
  e_label = NULL,
  text_size = 1,
  elabel_size = 5,
  e_size = 3,
  breaks_x = NULL,
  breaks_z = NULL
)
```

Arguments

RSA_object	An RSA object
model	The model to be probed for extrema (reversal or acceleration points)
acceleration	Rates of accelerations along the LOC and LOIC to be inspected (0< |rate| < 1). Passed on internally to ident.ext
n_sample	Number of random draws to consider to find extrema. This option is used for large samples to increase speed in preliminary analyses, but it is not recommended for published results). Passed on internally to ident.ext
names_xLOC	Label of x axis on the LOC plot.
names_xLOIC	Label of x axis on the LOIC plot.
names_z	Label of z axis (outcome).
xlim	Limits of the x axis
zlim	Limits of the z axis

e_label	If "none", no coordinates are projected. Defaults to NULL.
text_size	Text size for titles and labels.
elabel_size	Text size for extrema coordinates.
e_size	Point size of extrema.
breaks_x	Scale breaks for x axis. Defaults to seq(-10, 10, 0.5).
breaks_z	Scale breaks for z axis. Defaults to seq(-10, 10, 0.5).

Details

The lines of congruence (LOC) and incongruence (LOIC) are fundamental to RSA applications. This function allows plotting the response along the LOC and LOIC, thus offering a more precise visualization of the response surface (as a complement to 3d or 2d projections provided by RSAplot). The location of reversal or acceleration points can be provided with x-y-z coordinales or without them (e_label='none').

Value

A list of plot of the lines of congruence (LOC) and incongruence (LOIC)

See Also

RSAmodel

Examples

```
######ESTIMATE POLYNOMIAL MODEL FOR RSA
RSA_NSfit  <- RSAmodel(formula= engagement ~ needs*supplies,
data= sim_NSfit, model= c("FM26_PARALLELASYMWEAK"))
######PLOT EXTREMA OVER LOC AND LOIC
EXTsim <- plotting.ext(RSA_NSfit,model="FM26_PARALLELASYMWEAK",
xlim=c(-3,3),zlim=c(-3,3),acceleration=c(0,-0.3),
text_size=0.7,elabel_size=3,e_size=2,breaks_x=seq(-10, 10, 0.5),breaks_z=seq(-10, 10, 0.5))
ggpubr::ggarrange(EXTsim[["LOC"]], EXTsim[["LOIC"]],
labels = c("A. Response over LOC", "B. Response over LOIC"),
nrow=1,ncol=2,font.label = list(size = 11))
```

---

plotting.rsa *Plots the response surface of a polynomial model of first, second, or third degree*

Description

Plots the response surface of a model based on polynomial families (37 families), saturated polynomials, or user-specific polynomials.

Usage

```
plotting.rsaCOEF(
  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 = "CUBIC",
  acceleration = c(0, 0),
  FAST = TRUE,
  n_sample = 100,
  xlim = NULL,
  ylim = NULL,
  zlim = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  main = "",
  surface = "predict",
  lambda = NULL,
  suppress.surface = FALSE,
  suppress.box = FALSE,
  suppress.grid = FALSE,
  suppress.ticklabels = FALSE,
  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,
  coefs = FALSE,
 axes = c("LOC", "LOIC", "r1_LOC", "r2_LOC", "r1_LOIC", "r2_LOIC", "a1_LOC", "a2_LOC",
    "a1_LOIC", "a2_LOIC"),
  axesStyles = list(LOC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
   "black", "blue")), LOIC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
   "black", "blue")), PA1 = list(lty = "dotted", lwd = 2, col = ifelse(bw == TRUE,
  "black", "gray30")), PA2 = list(lty = "dotted", lwd = 2, col = ifelse(bw == TRUE,
  "black", "gray30")), r1_LOC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
  "black", "green")), r2_LOC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
```

```
    "black", "green")), r1_LOIC = list(lty = "solid",     lwd = 2, col = ifelse(bw ==
    TRUE, "black", "red")), r2_LOIC = list(lty = "solid", lwd = 2, col = ifelse(bw ==
    TRUE, "black", "red")), a1_LOC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
    TRUE, "black", "green")), a2_LOC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
    TRUE, "black", "green")), a1_LOIC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
    TRUE, "black", "red")), a2_LOIC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
      TRUE, "black", "red"))),
  project = c("contour"),
  maxlines = FALSE,
  cex.tickLabel = 1,
  cex.axesLabel = 1,
  cex.main = 1,
  points = list(data = NULL, n_random = NULL, show = NA, value = "raw", jitter = 0,
    color = "black", cex = 0.5, out.mark = FALSE),
  fit = NULL,
  link = "identity",
  tck = c(1.5, 1.5, 1.5),
  distance = c(1.3, 1.3, 1.4),
  border = FALSE,
  contour = list(show = FALSE, color = "grey40", highlight = c()),
  hull = NA,
  showSP = FALSE,
  showSP.CI = FALSE,
  pal = NULL,
  pal.range = "box",
  pad = 0,
  claxes.alpha = 0.05,
  demo = FALSE,
  ...
)

plotting.rsa(x, ...)
```

Arguments

x	Either an RSA object (returned by the RSAmodel 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)
x3	X^3 coefficient
xy2	XY^2 coefficient

x2y	X^2Y coefficient
y3	Y^3 coefficient
b0	Intercept
type	3d for 3d surface plot, 2d for 2d contour plot, "interactive" for interactive rotatable plot.
model	If x is an RSA object: from which model should the response surface be computed?
acceleration	Rates of accelerations along the LOC and LOIC to be inspected (0< |rate| < 1). Passed on internally to ident.ext
FAST	If FALSE, will also project response over LOC and LOIC. If TRUE, will only project the response surface (faster option).
n_sample	Number of random draws to consider to find extrema. This option is used for large samples to increase speed in preliminary analyses, but it is not recommended for published results). Passed on internally to ident.ext
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
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.
suppress.surface	Should the surface be suppressed (only for type="3d")? Useful for only showing the data points, or for didactic purposes (e.g., first show the cube, then fade in the surface).
suppress.box	Should the surrounding box be suppressed (only for type="3d")?
suppress.grid	Should the grid lines be suppressed (only for type="3d")?
suppress.ticklabels	Should the numbers on the axes be suppressed (only for type="3d")?
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?

`param` Should the surface parameters a1 to a5 be shown on the plot? In case of a 3d plot a1 to a5 are printed on top of the plot; in case of a contour plot the principal axes are plotted. Surface parameters are not printed for cubic surfaces.

`coefs` Should the regression coefficients b1 to b5 (b1 to b9 for cubic models) be shown on the plot? (Only for 3d plot)

`axes` *A vector of strings specifying the axes that should be plotted. Can be any combination of c("LOC", "LOIC","r1_LOC","r2_LOC","r1_LOIC","r2_LOIC","a1_LOC","a2_LOC","a1_L LOC = line of congruence, LOIC = line of incongruence, r1_LOC = first reversal point on LOC,r2_LOC = second reversal point on LOC,r1_LOIC = first reversal point on LOIC,r2_LOIC = second reversal point on LOIC,a1_LOC = first acceleration point on LOC,a2_LOC = second acceleration point on LOC,a1_LOIC = first acceleration point on LOIC,a2_LOIC = second acceleration point on LOIC.

`axesStyles` *Define the visual styles of the axes LOC,LOIC,r1_LOC,r2_LOC,r1_LOIC,r2_LOIC,a1_LOC,a2_LOC,a Provide a named list: `axesStyles=list(LOC = list(lty="solid", lwd=2, col=ifelse(bw==TRUE, "black", "blue"))`. It recognizes three parameters: `lty`, `lwd`, and `col`. If you define a style for an axis, you have to provide all three parameters, otherwise a warning will be shown.

`project` *A vector of graphic elements that should be projected on the floor of the cube. Can include any combination of c("LOC", "LOIC", "contour", "points"). Note that projected elements are plotted in the order given in the vector (first elements are plotted first and overplotted by later elements).

`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"

`cex.tickLabel` Font size factor for tick labels

`cex.axesLabel` Font size factor for axes labels

`cex.main` Factor for main title size

`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
- n_random *Number of randomly drawn data points to be plotted from data. Used to avoid cluttering in large datasets.
- show = TRUE: Should the original data points be overplotted?
- color = "black": Color of the points. Either a single value for all points, or a vector with the same size as data points provided. If parameter `fill` is also defined, `color` refers to the border of the points.
- fill = NULL: Fill of the points. Either a single value for all points, or a vector with the same size as data points provided. As a default, this is set to NULL, which means that all points simply have the color `color`.
- 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. Either a single value for all points, or a vector with the same size as data points provided.

- stilt: Should stilts be drawn for selected data points (i.e., lines from raw data points to the floor)? A logical vector with the same size as data points provided, indicating which points should get a stilt.
- 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: `RSAmodel(..., out.rm=TRUE)`.
  - If out.rm == TRUE (in RSAmodel()) and out.mark == FALSE (in plotting.rsa()), the outlier is removed from the model and *not plotted* in RSAplot.
  - If out.rm == TRUE (in RSAmodel()) and out.mark == TRUE (in plotting.rsa()), the outlier is removed from the model but plotted and marked in RSAplot.
  - If out.rm == FALSE (in RSAmodel()): Outliers are not removed and cannot be plotted.
  - Example syntax: `plotting.rsa(r1, points=list(show=TRUE, out.mark=TRUE))`

As a shortcut, you can also set `points=TRUE` to set the defaults.

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

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

`tck` A vector of three values defining the position of labels to the axes (see ?wireframe)

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

`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).

`showSP` Plot the stationary point? (only relevant for `type="contour"`)

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

`pal` A palette for shading.

`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)

`claxes.alpha`	Alpha level that is used to determine the axes K1 and K2 that demarcate the regions of significance for the cubic models "CL" and "RRCL"
`demo`	Do not change that parameter (internal use only)
`...`	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(getI "r2", model="CUBIC"), 3))))`

Details

This function plots the response surface in 3D or 2D. The function was adapted from the function plot.RSA (RSA package), by adding features for new polynomial families (+27 families, +user-specific polynomial) and new curvature probing tests (+accelerations, +LOIC extrema; Núñez-Regueiro & Juhel, 2022, 2024b). When curvatures are found, lines are projected that intersect inflection points (i.e., reversal or acceleration points) along the lines of congruence (x=y; in green) and incongruence (x=-y; in red), following derivatives of response curvatures (Núñez-Regueiro & Juhel, 2024a).

Value

A plot of the response surface

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

Núñez-Regueiro, F., Juhel, J. (2022). *Model-Building Strategies in Response Surface Analysis* Manuscript submitted for publication.

Núñez-Regueiro, F., Juhel, J. (2024a). *Response Surface Analysis for the Social Sciences I: Identifying Best-Fitting Polynomial Solutions* Manuscript submitted for publication.

Núñez-Regueiro, F., Juhel, J. (2024b). *Response Surface Analysis for the Social Sciences II: Combinatory Rationales for Complex Polynomial Models* Manuscript submitted for publication.

See Also

`plotting.ext`, `RSAmodel`

Examples

```
######PLOT RESPONSE SURFACE OF FM26_PARALLELASYMWEAK
PLOTsim3D <- plotting.rsa(RSA_step1,model="FM26_PARALLELASYMWEAK",type="3d",
acceleration=c(0,-0.3),points=list(show=TRUE, value="predicted"),
legend = FALSE,distance = c(1.2, 1.2, 1.2),cex.tickLabel = 0.6,
cex.axesLabel = 0.8,xlim=c(-3,3),ylim=c(-3,3),zlim=c(-3,3),hull=FALSE)
PLOTsim3D
```

plotting.rsaSIM *Plots the response surface of a simulated polynomial model*

Description

Plots the response surface of a model based on simulation parameter values.

Usage

```
plotting.rsaSIM(
  int = 0,
  x,
  y,
  x2,
  xy,
  y2,
  x3,
  x2y,
  xy2,
  y3,
  corr_xy = 0,
  e_label = NULL,
  center = "variablewise",
  scale = "variablewise",
  seed = 123,
  type = "3d",
  model = c("CUBIC", "QUADRATIC", "FM3_ADDITIVE"),
  acceleration = c(0, 0),
  FAST = TRUE,
  n_sample = 10000,
  xlim = c(-3, 3),
  ylim = c(-3, 3),
  zlim = c(-3, 3),
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  main = "",
  surface = "predict",
  lambda = NULL,
  suppress.surface = FALSE,
  suppress.box = FALSE,
  suppress.grid = FALSE,
  suppress.ticklabels = FALSE,
  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,
  coefs = FALSE,
 axes = c("LOC", "LOIC", "r1_LOC", "r2_LOC", "r1_LOIC", "r2_LOIC", "a1_LOC", "a2_LOC",
    "a1_LOIC", "a2_LOIC"),
  axesStyles = list(LOC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
   "black", "blue")), LOIC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
   "black", "blue")), PA1 = list(lty = "dotted", lwd = 2, col = ifelse(bw == TRUE,
  "black", "gray30")), PA2 = list(lty = "dotted", lwd = 2, col = ifelse(bw == TRUE,
  "black", "gray30")), r1_LOC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
  "black", "green")), r2_LOC = list(lty = "solid", lwd = 2, col = ifelse(bw == TRUE,
  "black", "green")), r1_LOIC = list(lty = "solid",     lwd = 2, col = ifelse(bw ==
  TRUE, "black", "red")), r2_LOIC = list(lty = "solid", lwd = 2, col = ifelse(bw ==
  TRUE, "black", "red")), a1_LOC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
  TRUE, "black", "green")), a2_LOC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
  TRUE, "black", "green")), a1_LOIC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
  TRUE, "black", "red")), a2_LOIC = list(lty = "twodash", lwd = 2, col = ifelse(bw ==
    TRUE, "black", "red"))),
  project = c("contour"),
  maxlines = FALSE,
  cex.tickLabel = 1,
  cex.axesLabel = 1,
  cex.main = 1,
  points = list(jitter = 0.1, show = T, value = "predicted"),
  fit = NULL,
  link = "identity",
  tck = c(1.5, 1.5, 1.5),
  distance = c(1.3, 1.3, 1.4),
  border = FALSE,
  contour = list(show = FALSE, color = "grey40", highlight = c()),
  hull = NA,
  showSP = FALSE,
  showSP.CI = FALSE,
  pal = NULL,
  pal.range = "box",
  pad = 0,
  claxes.alpha = 0.05,
  demo = FALSE,
  ...
)
```

Arguments

int	Intercept (b0)
x	X coefficient (b1)
y	Y coefficient (b2)
x2	X^2 coefficient (b3)
xy	XY interaction coefficient (b4)

y2	Y^2 coefficient (b5)
x3	X^3 coefficient (b9)
x2y	X^2Y coefficient (b7)
xy2	XY^2 coefficient (b8)
y3	Y^3 coefficient (b6)
corr_xy	Correlation between predictors (default to 0).
e_label	If "none", no extrema coordinates are projected. Defaults to NULL.
center	Method for centering the predictor variables before the analysis. Default option ("variablewise") centers the predictor variables on *their respective* sample mean. "none" applies no centering. "pooled" centers the predictor variables on their *pooled* sample mean. You should think carefully before applying the "pooled" option, as centering or reducing the predictor variables on common values (e.g., their grand means and SDs) can affect the commensurability of the predictor scales.
scale	Method for scaling the predictor variables before the analysis. Default option ("variablewise") scales the predictor variables on *their respective* sample SD. "none" applies no scaling. "pooled" scales the predictor variables on their *pooled* sample SD. You should think carefully before applying the "pooled" option, as scaling the predictor variables on common values (e.g., their grand SDs) can affect the commensurability of the predictor scales.
seed	Randomization seed for reproducible results
type	3d for 3d surface plot, 2d for 2d contour plot, "interactive" for interactive rotatable plot.
model	If x is an RSA object: from which model should the response surface be computed?
acceleration	Rates of accelerations along the LOC and LOIC to be inspected (0< |rate| < 1). Passed on internally to ident.ext
FAST	If FALSE, will also project response over LOC and LOIC. If TRUE, will only project the response surface (faster option).
n_sample	Size of simulated sample
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
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.
suppress.surface	Should the surface be suppressed (only for type="3d")? Useful for only showing the data points, or for didactic purposes (e.g., first show the cube, then fade in the surface).
suppress.box	Should the surrounding box be suppressed (only for type="3d")?
suppress.grid	Should the grid lines be suppressed (only for type="3d")?
suppress.ticklabels	Should the numbers on the axes be suppressed (only for type="3d")?
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?
param	Should the surface parameters a1 to a5 be shown on the plot? In case of a 3d plot a1 to a5 are printed on top of the plot; in case of a contour plot the principal axes are plotted. Surface parameters are not printed for cubic surfaces.
coefs	Should the regression coefficients b1 to b5 (b1 to b9 for cubic models) be shown on the plot? (Only for 3d plot)
axes	*A vector of strings specifying the axes that should be plotted. Can be any combination of c("LOC", "LOIC","r1_LOC","r2_LOC","r1_LOIC","r2_LOIC","a1_LOC","a2_LOC","a1_L LOC = line of congruence, LOIC = line of incongruence, r1_LOC = first reversal point on LOC,r2_LOC = second reversal point on LOC,r1_LOIC = first reversal point on LOIC,r2_LOIC = second reversal point on LOIC,a1_LOC = first acceleration point on LOC,a2_LOC = second acceleration point on LOC,a1_LOIC = first acceleration point on LOIC,a2_LOIC = second acceleration point on LOIC.
axesStyles	*Define the visual styles of the axes LOC,LOIC,r1_LOC,r2_LOC,r1_LOIC,r2_LOIC,a1_LOC,a2_LOC,a Provide a named list: axesStyles=list(LOC = list(lty="solid", lwd=2, col=ifelse(bw==TRUE, "black", "blue")). It recognizes three parameters: lty, lwd, and col. If you define a style for an axis, you have to provide all three parameters, otherwise a warning will be shown.
project	*A vector of graphic elements that should be projected on the floor of the cube. Can include any combination of c("LOC", "LOIC", "contour", "points"). Note that projected elements are plotted in the order given in the vector (first elements are plotted first and overplotted by later elements).
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"
cex.tickLabel	Font size factor for tick labels
cex.axesLabel	Font size factor for axes labels
cex.main	Factor for main title size
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
- n_random *Number of randomly drawn data points to be plotted from data. Used to avoid cluttering in large datasets.
- show = TRUE: Should the original data points be overplotted?
- color = "black": Color of the points. Either a single value for all points, or a vector with the same size as data points provided. If parameter `fill` is also defined, `color` refers to the border of the points.
- fill = NULL: Fill of the points. Either a single value for all points, or a vector with the same size as data points provided. As a default, this is set to NULL, which means that all points simply have the color `color`.
- 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. Either a single value for all points, or a vector with the same size as data points provided.
- stilt: Should stilts be drawn for selected data points (i.e., lines from raw data points to the floor)? A logical vector with the same size as data points provided, indicating which points should get a stilt.
- 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: `RSAmodel(..., out.rm=TRUE)`.
  - If out.rm == TRUE (in RSAmodel()) and out.mark == FALSE (in plotting.rsa()), the outlier is removed from the model and *not plotted* in RSAplot.
  - If out.rm == TRUE (in RSAmodel()) and out.mark == TRUE (in plotting.rsa()), the outlier is removed from the model but plotted and marked in RSAplot.
  - If out.rm == FALSE (in RSAmodel()): Outliers are not removed and cannot be plotted.
  - Example syntax: `plotting.rsa(r1, points=list(show=TRUE, out.mark=TRUE))`

As a shortcut, you can also set `points=TRUE` to set the defaults.

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

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

`tck` A vector of three values defining the position of labels to the axes (see ?wireframe)

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

`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).
showSP	Plot the stationary point? (only relevant for type="contour")
showSP.CI	Plot the CI of the stationary point? (only relevant for type="contour")
pal	A palette for shading.
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)
claxes.alpha	Alpha level that is used to determine the axes K1 and K2 that demarcate the regions of significance for the cubic models "CL" and "RRCL"
demo	Do not change that parameter (internal use only)
...	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(getI "r2", model="CUBIC"), 3))))

Details

Based on polynomial parameters provided by the user, this function simulates data and plots the corresponding response surface, including reversals and accelerations (when applicable).

Value

A plot of the simulated response surface

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

Núñez-Regueiro, F., Juhel, J. (2022). *Model-Building Strategies in Response Surface Analysis* Manuscript submitted for publication.

Núñez-Regueiro, F., Juhel, J. (2024a). *Response Surface Analysis for the Social Sciences I: Identifying Best-Fitting Polynomial Solutions* Manuscript submitted for publication.

Núñez-Regueiro, F., Juhel, J. (2024b). *Response Surface Analysis for the Social Sciences II: Combinatory Rationales for Complex Polynomial Models* Manuscript submitted for publication.

See Also

plotting.ext, RSAmodel

Examples

```
######SIMULATE RESPONSE SURFACE OF A MODEL FM26_PARALLELASYMWEAK (b9=1/3*b8)
###### WITH COR(X,Y)=.20
##Define polynomial parameters and correlation
b0 <- 0
b1 <- -0.064
b2 <- -b1*6
b3 <- -0.058
b5 <- b3*0.7
b4 <- 0
b6 <- 0
b7 <- 0
b8 <- -0.10
b9 <- 1/3*b8
corrXY <- 0.20
##Simulate corresponding surface
plotSIM <- plotting.rsaSIM(x=b1,y=b2,x2=b3,xy=b4,y2=b5,x3=b6,x2y=b7,xy2=b8,y3=b9,corr_xy= corrXY)
plotSIM
```

RSAmodel *Estimate polynomial models for response surface analysis*

Description

Estimates 37 polynomial families ("STEP1") or user-specific polynomial models ("USER") for use in response surface analysis, based on a 3-step identification strategy (Núñez-Regueiro & Juhel, 2022, 2024).

Usage

```
RSAmodel(
  formula,
  data = NULL,
  center = "none",
  scale = "none",
  na.rm = FALSE,
  out.rm = TRUE,
  breakline = FALSE,
  models = c("CUBIC", "STEP1"),
  user_model = NULL,
  verbose = TRUE,
  add = "",
  estimator = "MLR",
  se = "robust",
```

```
  missing = NA,
  control.variables = NULL,
  center.control.variables = FALSE,
  sampling.weights = NULL,
  group_name = NULL,
  cluster = NULL,
  ...
)
```

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	Method for centering the predictor variables before the analysis. Default option ("variablewise") centers the predictor variables on *their respective* sample mean. "none" applies no centering. "pooled" centers the predictor variables on their *pooled* sample mean. You should think carefully before applying the "pooled" option, as centering or reducing the predictor variables on common values (e.g., their grand means and SDs) can affect the commensurability of the predictor scales.
scale	Method for scaling the predictor variables before the analysis. Default option ("variablewise") scales the predictor variables on *their respective* sample SD. "none" applies no scaling. "pooled" scales the predictor variables on their *pooled* sample SD. You should think carefully before applying the "pooled" option, as scaling the predictor variables on common values (e.g., their grand SDs) can affect the commensurability of the predictor scales.
na.rm	Remove missings before proceeding?
out.rm	Should outliers according to Bollen & Jackman (1980) criteria be excluded from the 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
models	A vector with names of all models that should be computed. Should be any from c("CUBIC","FM1_ONLYX","FM2_ONLYY","FM3_ADDITIVE","FM4_INTER","FM5_QUADX","FM6_QUADY" For models="STEP1", all polynomial families and the saturated cubic are computed (default), for models="USER" all user-specific models defined in the list "user_model" are computed.
user_model	A list of user-specified polynomial models, defined by setting constraints on the polynomial parameters b1 to b9, using lavaan syntax (e.g., "b1 == 2*b2"). Only parametric constraints are allowed.
verbose	Should additional information during the computation process be printed?
add	Additional syntax that is added to the lavaan model. Can contain, for example, additional constraints, like "p01 == 0; p11 == 0"

`estimator` Type of estimator that should be used by lavaan. Defaults to "MLR", which provides robust standard errors, a robust scaled test statistic, and can handle missing values. If you want to reproduce standard OLS estimates, use `estimator="ML"` and `se="standard"`

`se` Type of standard errors. This parameter gets passed through to the `sem` function of the `lavaan` package. See options there. By default, robust SEs are computed. If you use `se="boot"`, `lavaan` provides CIs and p-values based on the bootstrapped standard error. If you use `confint(..., method="boot")`, in contrast, you get CIs and p-values based on percentile bootstrap.

`missing` Handling of missing values (this parameter is passed to the `lavaan sem` function). By default (`missing=NA`), Full Information Maximum Likelihood (FIML) is employed in case of missing values. If cases with missing values should be excluded, use `missing = "listwise"`.

`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.

`center.control.variables`
Should the control variables be centered before analyses? This can improve interpretability of the intercept, which will then reflect the predicted outcome value at the point (X,Y)=(0,0) when all control variables take their respective *average* values.

`sampling.weights`
Name of variable containing sampling weights. Needs to be added here (not in ...) to be included in the analysis dataset.

`group_name` Name of variable defining groups, for multigroup modeling.

`cluster` Name of variable for clusters, for cluster-level variance correction.

`...` Additional parameters passed to the `sem` function.

Details

This function implements a comparative framework for identifying best-fitting RSA solutions (Núñez-Regueiro & Juhel, 2022, 2024). The default feature ("STEP1") involves the comparison of 37 polynomial families predefined by parametric constraints, to identify likely candidates for best-fitting solution. Step 2 involves probing variants within the retained best-fitting family, by testing user-specific constraints ("USER") on lower-order polynomials that do not define the family. Step 3 (optional) can be conducted on the final variant by bootstrapping validation (using `bootstrapLavaan(RSA_object$models$name_final, FUN="coef")`, see examples) or cross-validation data.

Value

A list of objects containing polynomials models and names of variables

References

Núñez-Regueiro, F., Juhel, J. (2022). *Model-Building Strategies in Response Surface Analysis.* Manuscript submitted for publication.

Núñez-Regueiro, F., Juhel, J. (2024). *Response Surface Analysis for the Social Sciences I: Identifying Best-Fitting Polynomial Solutions.* Manuscript submitted for publication.

See Also

plotting.rsa, plotting.ext

Examples

```
###Simulation data from RSAtools
summary(sim_NSfit)

##### STEP 1: Estimate and compare polynomial families
##NB: to estimate all families, use "STEP1" in the "model" option
RSA_step1 <-  RSAmodel(engagement ~ needs*supplies,
data= sim_NSfit, model= c("CUBIC","FM20_ASYMCONG",
"FM21_ASYMINCONG","FM26_PARALLELASYMWEAK"))
names(RSA_step1$models)
##Compare solutions according to parsimony (wAIC=Akaike weight)
RSA_step1_fit <- best.rsa(RSA_step1,order=c("wAIC"))
head(RSA_step1_fit)
##Inspect best-fitting family model
summary(RSA_step1$models$FM26_PARALLELASYMWEAK)

##### STEP 2: Test variant within best-fitting family
##Define variant as constraints
list_variant <- list()
list_variant[["variant1"]] <- c('
####First-order polynomials: variant-specific
b1 == -1/4*b2
####Second-order polynomials: variant-specific
b5 == b3/2
b4 == 0
####Third-order polynomials: define family FM26
b6 == 0
b7 == 0
b9 == b8/-3
')
RSA_NSfit_Step2  <- RSAmodel(formula= engagement ~ needs*supplies,
data= sim_NSfit, model= c("FM26_PARALLELASYMWEAK","USER"),
user_model= list_variant)
##Compare variant to best-fitting family (e.g., LRT_pvalue p > .05)
best.rsa2(RSA_NSfit_Step2,m1="variant1",m2="FM26_PARALLELASYMWEAK")[2,1:3]

######PROBE RESPONSE SURFACE EXTREMA: see ident.ext()

######PLOT RESPONSE SURFACE: see plotting.rsa, plotting.ext
```

`RSAmodel.auxiliary` *Estimate a list of polynomial models for RSA, using auxiliary variables in FIML*

Description

Estimate any number of predefined or user-specific polynomial models, using auxiliary variables for missing data treatment by full information maximum likelihood (Graham, 2003). Based on the semTools function `sem.auxiliary`. Works in the same way as `RSAmodel`, only differing by the addition of the auxiliary variable option ("aux" option).

Usage

```
RSAmodel.auxiliary(
  formula,
  data = NULL,
  aux = NULL,
  center = "none",
  scale = "none",
  na.rm = FALSE,
  out.rm = TRUE,
  breakline = FALSE,
  models = c("CUBIC", "STEP1"),
  user_model = NULL,
  verbose = TRUE,
  add = "",
  estimator = "MLR",
  se = "robust",
  missing = NA,
  control.variables = NULL,
  center.control.variables = FALSE,
  sampling.weights = NULL,
  group_name = NULL,
  cluster = NULL,
  ...
)
```

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
`aux`	`character`. Names of auxiliary variables to add to `model`. Passed on internally to `sem.auxiliary` (semTools package)

`center` Method for centering the predictor variables before the analysis. Default option ("none") applies no centering. "pooled" centers the predictor variables on their *pooled* sample mean. "variablewise" centers the predictor variables on *their respective* sample mean. You should think carefully before applying the "variablewise" option, as centering or reducing the predictor variables on common values (e.g., their grand means and SDs) can affect the commensurability of the predictor scales.

`scale` Method for scaling the predictor variables before the analysis. Default option ("none") applies no scaling. "pooled" scales the predictor variables on their *pooled* sample SD, which preserves the commensurability of the predictor scales. "variablewise" scales the predictor variables on *their respective* sample SD. You should think carefully before applying the "variablewise" option, as scaling the predictor variables at different values (e.g., their respective SDs) can affect the commensurability of the predictor scales.

`na.rm` Remove missings before proceeding?

`out.rm` Should outliers according to Bollen & Jackman (1980) criteria be excluded from the 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`

`models` A vector with names of all models that should be computed. Should be any from `c("CUBIC","FM1_ONLYX","FM2_ONLYY","FM3_ADDITIVE","FM4_INTER","FM5_QUADX","FM6_QUADY"`
For `models="STEP1"`, all polynomial families and the saturated cubic are computed (default), for `models="USER"` all user-specific models defined in the list "user_model" are computed.

`user_model` A list of user-specified polynomial models, defined by setting constraints on the polynomial parameters b1 to b9, using the syntax in lavaan, for example: "b1 == 2*b2". Only parametric constraints specifications are allowed.

`verbose` Should additional information during the computation process be printed?

`add` Additional syntax that is added to the lavaan model. Can contain, for example, additional constraints, like "p01 == 0; p11 == 0"

`estimator` Type of estimator that should be used by lavaan. Defaults to "MLR", which provides robust standard errors, a robust scaled test statistic, and can handle missing values. If you want to reproduce standard OLS estimates, use `estimator="ML"` and `se="standard"`

`se` Type of standard errors. This parameter gets passed through to the `sem` function of the `lavaan` package. See options there. By default, robust SEs are computed. If you use `se="boot"`, `lavaan` provides CIs and p-values based on the bootstrapped standard error. If you use `confint(..., method="boot")`, in contrast, you get CIs and p-values based on percentile bootstrap.

`missing` Handling of missing values (this parameter is passed to the `lavaan sem` function). By default (`missing=NA`), Full Information Maximum Likelihood (FIML) is employed in case of missing values. If cases with missing values should be excluded, use `missing = "listwise"`.

`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.

`center.control.variables`
: Should the control variables be centered before analyses? This can improve interpretability of the intercept, which will then reflect the predicted outcome value at the point (X,Y)=(0,0) when all control variables take their respective *average* values.

`sampling.weights`
: Name of variable containing sampling weights. Needs to be added here (not in ...) to be included in the analysis dataset.

`group_name`
: Name of variable defining groups, for multigroup modeling.

`cluster`
: Name of variable for clusters, for cluster-level variance correction.

`...`
: Additional parameters passed to the `sem.auxiliary` function.

Details

This function implements a comparative framework for identifying best-fitting RSA solutions (Núñez-Regueiro & Juhel, 2022, 2024). The default feature ("STEP1") involves the comparison of 37 polynomial families predefined by parametric constraints (against 10 families in the RSA package), to identify likely candidates for best-fitting solution. Step 2 involves probing variants within the retained best-fitting family, by testing user-specific constraints ("USER") on lower-order polynomials that do not define the family. Step 3 can be conducted on the final variant by parametric bootstrapping using `bootstrapLavaan(RSA_object$models$name_final, FUN="coef")` or cross-validation data.

Value

A list of objects containing polynomials models and names of variables

References

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling, 10*(1), 80-100, DOI:10.1207/S15328007SEM1001_4

Núñez-Regueiro, F., Juhel, J. (2022). *Model-Building Strategies in Response Surface Analysis* Manuscript submitted for publication.

Núñez-Regueiro, F., Juhel, J. (2024). *Response Surface Analysis for the Social Sciences I: Identifying Best-Fitting Polynomial Solutions* Manuscript submitted for publication.

See Also

`RSAmodel`, `sem.auxiliary`

RSA_step1 *Simulation data on needs-supplies fit processes (STEP1)*

Description

An object of class "RSA" containing estimates of all polynomial families, obtained by passing the function `RSAmodel(formula= engagement ~ needs*supplies,data= sim_NSfit, model= c("CUBIC","STEP1"))`. Used to illustrate functions.

Author(s)

Fernando Núñez-Regueiro <fernando.nr.france@gmail.com>

sim_NSfit *Simulation data on needs-supplies fit processes (illustration)*

Description

Simulation data on needs-supplies fit processes in the context of student engagement at school (N= 500). For example, this could relate to needs and supplies in teacher autonomy support (Núñez-Regueiro et al., 2024, 2025) or in school environment characteristics (Fraser & Rentoul, 1980). The data was simulated to reflect a parallel asymmetric weak congruence and strong incongruence effect (polynomial family 26; Núñez-Regueiro et al., 2022, 2024c)

Author(s)

Fernando Núñez-Regueiro <fernando.nr.france@gmail.com>

References

Fraser, B.J., Rentoul, A.J. (1980). Person-environment fit in open classrooms.*The Journal of Educational Research*, 73(3), 159-167. https://doi.org/10.1080/00220671.1980.10885227

Núñez-Regueiro, F., Juhel, J. (2022). *Model-Building Strategies in Response Surface Analysis.* Manuscript submitted for publication.

Núñez-Regueiro, F., Santana-Monagas, E., Juhel, J. (2024).How Needs-Supplies Fit With Teachers, Peers, and Parents Relate to Youth Outcomes.*Journal of Youth and Adolescence*, https://doi.org/10.1007/s10964-024-02049-9

Núñez-Regueiro, F., Juhel, J., Wang, M-T. (2025). Does Needs Satisfaction Reflect Positive Needs-Supplies Fit or Misfit? A New Look at Autonomy Supportive Contexts Using Cubic Response Surface Analysis.*Social Psychology of Education*, https://doi.org/10.1007/s11218-024-09959-3.

Núñez-Regueiro, F., Juhel, J. (2024c). *Response Surface Analysis for the Social Sciences I: Identifying Best-Fitting Polynomial Solutions.* Manuscript submitted for publication.

Index