

# Package ‘RSA’

July 21, 2025

**Encoding** UTF-8

**Type** Package

**Title** Response Surface Analysis

**Version** 0.10.6

**Date** 2023-01-12

**Maintainer** Felix Schönbrodt <felix@nicebread.de>

**Description** Advanced response surface analysis. The main function `RSA` computes and compares several nested polynomial regression models (full second- or third-order polynomial, shifted and rotated squared difference model, rising ridge surfaces, basic squared difference model, asymmetric or level-dependent congruence effect models). The package provides plotting functions for 3d wireframe surfaces, interactive 3d plots, and contour plots. Calculates many surface parameters (`a1` to `a5`, principal axes, stationary point, eigenvalues) and provides standard, robust, or bootstrapped standard errors and confidence intervals for them.

**Suggests** fields, rgl, qgraph, tcltk, tkrplot, testthat, covr

**Depends** R (>= 2.15.0), lavaan (>= 0.5.20), ggplot2, lattice

**Imports** plyr, RColorBrewer, aplpack, methods

**License** GPL (>= 2)

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Felix Schönbrodt [cre, aut],  
Sarah Humberg [aut]

**Repository** CRAN

**Date/Publication** 2023-01-12 08:20:12 UTC

## Contents

|                      |   |
|----------------------|---|
| <code>aictab</code>  | 2 |
| <code>caRange</code> | 4 |
| <code>clRange</code> | 5 |

|                         |           |
|-------------------------|-----------|
| compare . . . . .       | 6         |
| compare2 . . . . .      | 6         |
| confint.RSA . . . . .   | 7         |
| demoRSA . . . . .       | 8         |
| fitted.RSA . . . . .    | 11        |
| getPar . . . . .        | 11        |
| modeltree . . . . .     | 12        |
| motcon . . . . .        | 13        |
| motcon2 . . . . .       | 14        |
| movieRSA . . . . .      | 14        |
| plotRSA . . . . .       | 16        |
| residuals.RSA . . . . . | 22        |
| RSA . . . . .           | 22        |
| RSA.ST . . . . .        | 27        |
| selfother . . . . .     | 29        |
| <b>Index</b>            | <b>31</b> |

---

|        |  |
|--------|--|
| aictab | <i>Show a table of AIC model comparisons</i> |
|--------|--|

---

**Description**

Show a table of AIC model comparisons

**Usage**

```
aictab(  
  x,  
  plot = FALSE,  
  bw = FALSE,  
  models = names(x$models)[!names(x$models) %in% c("absdiff", "absunc")],  
  digits = NA  
)
```

**Arguments**

|        |  |
|--------|--|
| x      | An RSA object  |
| plot   | Should a plot of the AICc table be plotted?  |
| bw     | Should the plot be black & white?  |
| models | A vector with all model names of the candidate set. Defaults to all polynomial models in the RSA object. |
| digits | The output is rounded to this number of digits. No rounding if NA (default).                             |

**Value**

**Modnames** Model names.

**K** Number of estimated parameters (including the intercept, residual variance, and, if present in the model, control variables).

**LL** Model log-likelihood.

**AICc** Akaike Information Criterion (corrected).

**Delta\_AICc** Difference in AICc between this model and the best model.

**AICcWt** The Akaike weights, also termed "model probabilities" by Burnham and Anderson (2002). Indicates the level of support (i.e., weight of evidence) of a model being the most parsimonious among the candidate model set.

**Cum.Wt** Cumulative Akaike weight. One possible strategy is to restrict interpretation to the "confidence set" of models, that is, discard models with a Cum.Wt > .95 (see Burnham & Anderson, 2002, for details and alternatives).

**evidence.ratio** Likelihood ratio of this model vs. the best model.

**cfi** Comparative Fit Index (CFI).

**R2** Coefficient of determination (R-squared).

**R2.adj** Adjusted R-squared.

**R2.baseline** Only provided if the model contains control variables. Difference in R-squared as compared to the baseline model with intercept and control variables (= the model "null"). This  $R^2$  increment will typically be of interest because it refers to the amount of variance explained by the two predictors X and Y (plus their squared and interaction terms) in the RSA model.

**R2.baseline.p** Only provided if the model contains control variables. p-value for the F-test of the model against the baseline model.

**Note**

This function is similar to the function aictab in the AICcmodavg package.

**References**

Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach*. Springer Science & Business Media.

**Examples**

```
## Not run:
data(motcon)
r.m <- RSA(postVA~ePow*iPow, motcon, verbose=FALSE)
aictab(r.m, plot=TRUE)

## End(Not run)
```

---

caRange

Range check for the CA/RRCA model

---

## Description

Identify data points behind E2 and test how many of them have outcome predictions that significantly differ from predictions for predictor combinations on E2 (that have the same level)

## Usage

```
caRange(
  object,
  alpha = 0.05,
  verbose = TRUE,
  model = "CA",
  alphacorrection = "none"
)
```

## Arguments

|                 |   |
|-----------------|---|
| object          | An RSA object   |
| alpha           | Alpha level for the one-sided confidence interval of the outcome predictions on E2  |
| verbose         | Should extra information be printed?  |
| model           | Either "CA" or "RRCA"   |
| alphacorrection | Set "Bonferroni" to adjust the alpha level for multiple testing when testing the outcome predictions of all data points behind E2 |

## Details

When testing an asymmetric congruence hypothesis with the CA or RRCA model, the caRange function helps to determine whether the hypothesis is supported for the whole range of realistic predictor combinations. It computes the position of the second extremum line E2 and tests how many predictor combinations are in the data which lie "behind" this line and, at the same time, have a significantly different outcome prediction than points on E2.

When plotting the estimated model (CA or RRCA) with [plot](#), you can plot the line E2 and the surface above this line by calling "E2" in the options project and axes.

## References

Humberg, S., Schönbrodt, F. D., Back, M. D., Nestler, S. (in preparation). *Cubic response surface analysis: Investigating asymmetric and level-dependent congruence effects with third-order polynomial models*. Manuscript submitted for publication.

---

|         |  |
|---------|--|
| clRange | <i>Range check for the CL/RRCL model</i> |
|---------|--|

---

## Description

Compute the regions of significance and test their intersection with the data

## Usage

```
clRange(object, alpha = 0.05, verbose = TRUE, model = "CL")
```

## Arguments

|         |  |
|---------|--|
| object  | An RSA object  |
| alpha   | Alpha level for the regions of significance of the surface's curvature |
| verbose | Should extra information be printed?                                   |
| model   | Either "CL" or "RRCL"  |

## Details

When testing a level-dependent congruence hypothesis with the CL or RRCL model, the `clRange` function helps to determine whether the hypothesis is supported for the whole range of realistic predictor combinations. It computes the mean predictor levels `k1` and `k2` at which the curvature of the surface changes its significance status. For each of the resulting intervals, the function informs whether the curvature is significantly negative, nonsignificant, or significantly positive in the respective interval.

When plotting the estimated model (CL or RRCL) with `plot`, you can plot the lines at which the significance status of the curvature changes and the surface above these lines by calling "K1" and "K2" in the options `project` and `axes`.

## References

Humberg, S., Schönbrodt, F. D., Back, M. D., Nestler, S. (in preparation). *Cubic response surface analysis: Investigating asymmetric and level-dependent congruence effects with third-order polynomial models*. Manuscript submitted for publication.

---

|         |  |
|---------|--|
| compare | <i>Compare a full list of RSA models</i> |
|---------|--|

---

**Description**

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

**Usage**

```
compare(x, verbose = TRUE, plot = FALSE, digits = 3, ...)
```

**Arguments**

|         |  |
|---------|--|
| x       | An RSA object  |
| verbose | Should the summary be printed?   |
| plot    | Should the comparison be plotted (using the <a href="#">modeltree</a> function)? |
| digits  | Digits of the output   |
| ...     | Additional parameters passed to the <a href="#">modeltree</a> function           |

**Details**

No details so far.

---

|          |  |
|----------|--|
| compare2 | <i>Compare two specific RSA models</i> |
|----------|--|

---

**Description**

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

**Usage**

```
compare2(x, m1 = "", m2 = "full", digits = 3, verbose = TRUE)
```

**Arguments**

|         |                                |
|---------|--------------------------------|
| x       | An RSA object                  |
| m1      | Name of first model            |
| m2      | Name of second model           |
| digits  | Digits of the output           |
| 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, and R2 indices, but *\*not\** for the chi2-LR test!

---

|             |   |
|-------------|---|
| confint.RSA | <i>Computes confidence intervals for RSA parameters, standard or bootstrapped</i> |
|-------------|---|

---

## Description

Computes confidence intervals for RSA parameters, standard or bootstrapped (using a percentile bootstrap)

## 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.   |
| ...    | Additional parameters passed to the bootstrapLavaan function, e.g., parallel="multicore", ncpus=2.                   |
| model  | A string specifying the model; defaults to "full"  |
| digits | Number of digits the output is rounded to; if NA, digits are unconstrained   |
| 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  |

## Details

There are two ways of getting bootstrapped CIs and p-values in the RSA package. If you use the option `se="boot"` in the [RSA](#) function, lavaan provides CIs and p-values based on the bootstrapped standard error (*not* percentile bootstraps). If you use `confint(..., method="boot")`, in contrast, you get CIs and p-values based on percentile bootstrap.

**See Also**[RSA](#)**Examples**

```
## 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.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))
# 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),
xlab = NULL,
ylab = NULL,
zlab = NULL,
points = TRUE,
model = "full",
project = c("PA1", "PA2"),
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 <sup>2</sup> coefficient   |
| y2   | Y <sup>2</sup> 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 <sup>3</sup> coefficient   |
| xy2  | XY <sup>2</sup> coefficient  |
| x2y  | X <sup>2</sup> Y coefficient   |
| y3   | Y <sup>3</sup> coefficient   |
| b0   | Intercept  |
| 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 |
| zlim | Limits of the z axis   |
| xlim | Limits of the x axis   |
| ylim | Limits of the y axis   |
| xlab | Label of the x axis  |
| ylab | Label of the y axis  |
| zlab | Label of the z axis  |

|          |   |
|----------|---|
| 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. See ?plotRSA for details. |
| model    | If x is an RSA object: from which model should the response surface be computed?  |
| project  | Which features should be projected on the floor? See ?plotRSA for details.  |
| 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 | <i>Return fitted values of a RSA model</i> |
|------------|--|

---

### 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 | <i>Retrieves several variables from an RSA object</i> |
|--------|---|

---

### 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", "p.value"     |
| 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 <a href="#">compare</a> 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](#)

**Examples**

```
## Not run:
data(motcon)
r.m <- RSA(postVA~ePow*iPow, motcon)
c1 <- compare(r.m)
modeltree(c1)

## End(Not run)
```

---

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 'ungluecklich ... gluecklich' (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

---

 motcon2

*Another data set on motive congruence.*


---

### Description

A dataset containing the explicit intimacy motive, implicit affiliation/intimacy motive and self ratings of affective valence. The variables are as follows:

### Format

A data frame with 362 rows and 3 variables

### Details

- EM Explicit intimacy motive, measured with a questionnaire (Unified Motive Scales, Schönbrodt & Gerstenberg, 2012). Raw values have been z standardized.
- IM Implicit affiliation/intimacy motive, measured with picture story exercise (6 pictures). Raw motive scores have been controlled for word count and z standardized.
- VA z standardized valence rating. Consists of two bipolar items from the PANAVA questionnaire (Schallberger, 2005): 'zufrieden ... unzufrieden' (satisfied ... unsatisfied) and 'ungluecklich ... gluecklich' (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 <a href="#">plotRSA</a> for details. |
| dur        | Duration of the movie in milliseconds   |
| 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,
  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", "PA1", "PA2"),
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"))),
project = c("contour"),
maxlines = FALSE,
cex.tickLabel = 1,
cex.axesLabel = 1,
cex.main = 1,
points = list(data = NULL, show = NA, value = "raw", jitter = 0, color = "black", cex =
  0.5, stilt = NULL, out.mark = FALSE, fill = NULL),
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

|    |   |
|----|---|
| x  | Either an RSA object (returned by the RSA function), or the coefficient for the X predictor |
| y  | Y coefficient   |
| x2 | X <sup>2</sup> coefficient  |
| y2 | Y <sup>2</sup> 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 <sup>3</sup> coefficient  |
| xy2                 | XY <sup>2</sup> coefficient   |
| x2y                 | X <sup>2</sup> Y coefficient  |
| y3                  | Y <sup>3</sup> coefficient  |
| b0                  | Intercept   |
| 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  |
| model               | If x is an RSA object: from which model should the response surface be computed?  |
| 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 labels (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)  |

|                            |   |
|----------------------------|---|
| <code>axes</code>          | A vector of strings specifying the axes that should be plotted. Can be any combination of <code>c("LOC", "LOIC", "PA1", "PA2", "E2", "K1", "K2")</code> . LOC = line of congruence, LOIC = line of incongruence, PA1 = first principal axis, PA2 = second principal axis, E2 = second extremum line in the CA or RRCA model, K1, K2 = boundary lines of the regions of significance in the CL or RRCL model.  |
| <code>axesStyles</code>    | Define the visual styles of the axes LOC, LOIC, PA1, PA2, E2, K1, and K2. Provide a named list: <code>axesStyles=list(LOC = list(lty="solid", lwd=2, col=ifelse(bw==TRUE, "black", "blue")))</code> . It recognizes three parameters: <code>lty</code> , <code>lwd</code> , and <code>col</code> . If you define a style for an axis, you have to provide all three parameters, otherwise a warning will be shown.  |
| <code>project</code>       | A vector of graphic elements that should be projected on the floor of the cube. Can include any combination of <code>c("LOC", "LOIC", "PA1", "PA2", "contour", "points", "E2", "K1", "K2")</code> . Note that projected elements are plotted in the order given in the vector (first elements are plotted first and overplotted by later elements).   |
| <code>maxlines</code>      | Should the maximum lines be plotted? (red: maximum X for a given Y, blue: maximum Y for a given X). Works only in <code>type="3d"</code>  |
| <code>cex.tickLabel</code> | Font size factor for tick labels  |
| <code>cex.axesLabel</code> | Font size factor for axes labels  |
| <code>cex.main</code>      | Factor for main title size  |
| <code>points</code>        | A list of parameters which define the appearance of the raw scatter points: <ul style="list-style-type: none"> <li>• <code>data</code>: 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</li> <li>• <code>show = TRUE</code>: Should the original data points be overplotted?</li> <li>• <code>color = "black"</code>: Color of the points. Either a single value for all points, or a vector with the same size as data points provided. If parameter <code>fill</code> is also defined, <code>color</code> refers to the border of the points.</li> <li>• <code>fill = NULL</code>: 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 <code>NULL</code>, which means that all points simply have the color <code>color</code>.</li> <li>• <code>value="raw"</code>: Plot the original z value, <code>"predicted"</code>: plot the predicted z value</li> <li>• <code>jitter = 0</code>: Amount of jitter for the raw data points. For z values, a value of 0.005 is reasonable</li> <li>• <code>cex = .5</code>: multiplication factor for point size. Either a single value for all points, or a vector with the same size as data points provided.</li> <li>• <code>stilt</code>: 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.</li> <li>• <code>out.mark = FALSE</code>: If set to <code>TRUE</code>, outliers according to Bollen &amp; Jackman (1980) are printed as red X symbols, but only when they have been removed in the RSA function: <code>RSA(..., out.rm=TRUE)</code>. <ul style="list-style-type: none"> <li>– If <code>out.rm == TRUE</code> (in <code>RSA()</code>) and <code>out.mark == FALSE</code> (in <code>plotRSA()</code>), the outlier is removed from the model and <i>*not plotted*</i> in <code>plotRSA</code>.</li> </ul> </li> </ul> |

- 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))`

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

|                           |  |
|---------------------------|--|
| <code>fit</code>          | Do not change that parameter (internal use only)   |
| <code>link</code>         | Link function to transform the z axes. Implemented are "identity" (no transformation; default), "probit", and "logit"  |
| <code>tck</code>          | A vector of three values defining the position of labels to the axes (see <code>?wireframe</code> )  |
| <code>distance</code>     | A vector of three values defining the distance of labels to the axes   |
| <code>border</code>       | 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 <code>border=FALSE</code> .  |
| <code>contour</code>      | A list defining the appearance of contour lines (aka. height lines). <code>show=TRUE</code> : Should the contour lines be plotted on the 3d wireframe plot? (Parameter only relevant for <code>type="3d"</code> ). <code>color = "grey40"</code> : Color of the contour lines. <code>highlight = c()</code> : 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. |
| <code>hull</code>         | 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).   |
| <code>showSP</code>       | Plot the stationary point? (only relevant for <code>type="contour"</code> )  |
| <code>showSP.CI</code>    | Plot the CI of the stationary point? (only relevant for <code>type="contour"</code> )  |
| <code>pal</code>          | A palette for shading. You can use <code>colorRampPalette</code> to construct a color ramp, e.g. <code>plot(r.m, pal=colorRampPalette(c("darkgreen", "yellow", "darkred"))(20))</code> . If <code>pal="flip"</code> , the default palette is used, but reversed (so that red is on top and green on the bottom).   |
| <code>pal.range</code>    | Should the color range be scaled to the box ( <code>pal.range = "box"</code> , default), or to the min and max of the surface ( <code>pal.range = "surface"</code> )? If set to "box", different surface plots can be compared along their color, as long as the <code>zlim</code> is the same for both.   |
| <code>pad</code>          | Pad controls the margin around the figure (positive numbers: larger margin, negative numbers: smaller margin)  |
| <code>claxes.alpha</code> | 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"   |
| <code>demo</code>         | Do not change that parameter (internal use only)   |
| <code>...</code>          | Additional parameters passed to the plotting function (e.g., <code>sub="Title"</code> ). A useful title might be the R squared of the plotted model: <code>sub = as.expression(bquote(R^2==.(round(getr2", model="full"), 3))))</code>   |

## 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 [RSA](#) 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
## Not run:
# 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"))

## End(Not run)
```

---

|               |  |
|---------------|--|
| residuals.RSA | <i>Return residual values of a RSA model</i> |
|---------------|--|

---

### 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 | <i>Performs several RSA model tests on a data set with two predictors</i> |
|-----|---|

---

### Description

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

### Usage

```
RSA(
  formula,
  data = NULL,
  center = "none",
  scale = "none",
  na.rm = FALSE,
  out.rm = TRUE,
  breakline = FALSE,
  models = "default",
  cubic = FALSE,
  verbose = TRUE,
  add = "",
  estimator = "MLR",
  se = "robust",
```

```

missing = NA,
control.variables = c(),
center.control.variables = FALSE,
...
)

```

## Arguments

|           |  |
|-----------|--|
| formula   | A formula in the form $z \sim x*y$ , 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 ("none") applies no centering. "pooled" centers the predictor variables on their <i>pooled</i> sample mean, which preserves the commensurability of the predictor scales. "variablewise" centers the predictor variables on <i>their respective</i> sample mean. You should think carefully before applying the "variablewise" option, as centering the predictor variables at different values (e.g., their respective means) 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 <i>pooled</i> sample SD, which preserves the commensurability of the predictor scales. "variablewise" scales the predictor variables on <i>their respective</i> 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 <code>c("absdiff", "absunc", "diff", "mean", "additive", "IA", "SQD", "RR", "SRR", "SRRR", "SSQD", "SRSQD", "full", "null", "onlyx", "onlyy", "onlyx2", "onlyy2", "cubic", "CA", "RRCA", "CL", "RRCL")</code> . For <code>models="all"</code> , all models are computed, for <code>models="default"</code> all models besides absolute difference models are computed.   |
| cubic     | Should the cubic models with the additional terms $Y^3$ , $XY^2$ , $YX^2$ , and $X^3$ be included?   |
| 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 <code>"p01 == 0; p11 == 0"</code>  |

|                          |  |
|--------------------------|--|
| 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 (see also <a href="#">confint.RSA</a> ). |
| 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 <i>average</i> values.   |
| ...                      | Additional parameters passed to the lavaan <a href="#">sem</a> function.   |

## Details

Even if the main variables of the model are normally distributed, their squared terms and interaction terms are necessarily non-normal. By default, the RSA function uses a scaled test statistic (test="Satorra-Bentler") and robust standard errors (se="robust"), which are robust against violations of the normality assumption.

*Why does my standard polynomial regression give different p-values and SEs than the RSA package? Shouldn't they be the same?* This is due to the robust standard errors employed in the RSA package. If you set estimator="ML" and se="standard", you get p-values that are very close to the standard approach. (They might still not be identical because the standard regression approach usually uses an OLS estimator and RSA uses an ML estimator).

Experimental feature (use with caution!): You can also fit **binary outcome variables** with a probit link function. For that purpose, the response variable has to be defined as "ordered", and the lavaan estimator changed to "WLSMV": `r1.binary <- RSA(z.binary~x*y, df, ordered="z.binary", estimator="WLSMV", model="full")` (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.binary, link="probit", xlim=c(0, 1), zlab="Probability")`. For plotting, the binary outcome variable must be coded with 0 and 1 (not as a factor). lavaan at the moment only supports a probit link function for binary outcomes, not a logit link. Please be aware that this experimental feature can fit the full model, but most other functions (such as model comparisons) might break and errors might show up.



## Note

For explanations of the meaning of the various different models that can be estimated, please see Schönbrodt (2016) for the second-order models (i.e., all models but "CA", "RRCA", "CL", "RRCL") and Humberg et al. (in press) for the third-order (cubic) models ("CA", "RRCA", "CL", "RRCL").

For most of the second-order models, several auxiliary parameters are computed from the estimated model coefficients (e.g.,  $a_1$ , ...,  $a_5$ ,  $p_{10}$ ,  $p_{11}$ ,  $p_{20}$ ,  $p_{21}$ ) and printed in the summary output. They can be used to guide interpretation by means of response surface methodology. Some references that explain how to use these parameters for interpretation are Edwards (2002; comprehensive overview of response surface methodology), Humberg et al. (2019; interpretation of  $a_1$ ,  $a_2$ ,  $a_3$ ,  $a_4$ ,  $p_{10}$ , and  $p_{11}$ , and how to use them to investigate congruence effects), Nestler et al. (2019; interpretation of  $a_1$ ,  $a_2$ ,  $a_3$ ,  $a_4$ , and  $a_5$ , and how to use them to investigate congruence effects, see in particular Appendix A for the introduction of  $a_5$ ), and Schönbrodt et al. (2018; interpretation of  $a_1$ , ...,  $a_5$ , see in particular Appendix A for  $a_5$ ).

The print function provides descriptive statistics about discrepancies in the predictors (with respect to numerical congruence). A cutpoint of  $|\Delta z| > 0.5$  is used. The computation generally follows the idea of Shannock et al (2010) and Fleenor et al. (1996). However, in contrast to them, we standardize to the common mean and the common SD of both predictor variables. Otherwise we would break commensurability, and a person who has  $x=y$  in the unstandardized variable could become incongruent after variable-wise standardization. See also our discussion of commensurability and scale transformation in the cubic RSA paper (Humberg et al., in press; see pp. 35 - 37 in the preprint at <https://psyarxiv.com/v6m35>).

## References

- Edwards, J. R. (2002). Alternatives to difference scores: Polynomial regression analysis and response surface methodology. In F. Drasgow & N. W. Schmitt (Eds.), *Advances in measurement and data analysis* (pp. 350–400). San Francisco, CA: Jossey-Bass.
- Humberg, S., Nestler, S., & Back, M. D. (2019). Response Surface Analysis in Personality and Social Psychology: Checklist and Clarifications for the Case of Congruence Hypotheses. *Social Psychological and Personality Science*, 10(3), 409–419. doi:10.1177/1948550618757600
- Humberg, S., Schönbrodt, F. D., Back, M. D., & Nestler, S. (in press). Cubic response surface analysis: Investigating asymmetric and level-dependent congruence effects with third-order polynomial models. *Psychological Methods*. doi:10.1037/met0000352
- Nestler, S., Humberg, S., & Schönbrodt, F. D. (2019). Response surface analysis with multilevel data: Illustration for the case of congruence hypotheses. *Psychological Methods*, 24(3), 291–308. doi:10.1037/met0000199
- Schönbrodt, F. D. (2016). *Testing fit patterns with polynomial regression models*. Retrieved from [osf.io/3889z](https://osf.io/3889z)
- Schönbrodt, F. D., Humberg, S., & Nestler, S. (2018). Testing similarity effects with dyadic response surface analysis. *European Journal of Personality*, 32(6), 627–641. doi:10.1002/per.2169

## See Also

[demoRSA](#), [plotRSA](#), [RSA.ST](#), [confint.RSA](#), [compare](#)

## 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)
})
df$z.binary <- as.numeric(df$z.sq < median(df$z.sq))

## Not run:
r1 <- RSA(z.sq~x*y, df)
summary(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

# Example with binary outcome
(probit regression, see Details; Experimental and a dirty workaround!).
# The standard summary output does not work; you have to access the full model directly:
r1.binary <- RSA(z.binary~x*y, df, ordered="z.binary", estimator="WLSMV",
  model="full", se="standard")
# --> ignore the warning
summary(r1.binary$models[["full"]])
plot(r1.binary, link="probit", xlim=c(0, 1), zlab="Probability")

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

# Get bootstrapped 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")

# Inclusion of control variables: Fake data on self-other agreement
```

```

data(selfother)
r.c <- RSA(liking~IQ_self*IQ_friend,
           center="pooled",
           control.variables=c("age", "int"),
           center.control.variables = TRUE,
           data=selfother)
summary(r.c)

## 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 = 0,
  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 <sup>2</sup> coefficient  |
| xy | XY interaction coefficient  |
| y2 | Y <sup>2</sup> 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 with exactly five elements with the names of the coefficients, e.g.: SE=c(x=.1, y=.2, x2=.1, y2=.5, xy=.3). SEs of all parameters have to be provided, otherwise the function will print an error. In case standard errors <i>and</i> the covariances (see below) <i>and</i> 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 with exactly four elements with the names of four specific covariances, e.g.: COV=c(x_y=.1, x2_y2=.2, x2_xy=.3, y2_xy=.4), where x_y is the covariance between x and y, and so on. All these covariances have to be 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 second-order polynomial model, this is 'n - 6 - number of control variables' in a regular regression (the following parameters are estimated: Intercept, x, y, x2, xy, y2, all control variables). df should be a single number.  |
| 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 a5. If an RSA object or SE, COV and df are provided, also significance test and standard errors of a1 to a5 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.curvX: quadratic component of the first PA, as seen from X axis PA2.curvX: quadratic component of the second PA, as seen from X axis PA1.curv: quadratic component of the first PA, after optimal coord transformation PA2.curv: quadratic component of the second PA, after optimal coord transformation

### References

Shanock, L. R., Baran, B. E., Gentry, W. A., Pattison, S. C., & Heggstad, 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 Shanock, L. R., Baran, B. E., Gentry, W. A., & Pattison, S. C. (2014). Erratum to: Polynomial regression with response surface analysis: A powerful approach for examining moderation and overcoming limitations of difference scores. *Journal of Business and Psychology*, 29, . <http://doi.org/10.1007/s10869-013-9317-6>

### 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)

## Compute standard errors and p values for surface parameters
## from external regression coefficients:
# standard errors for coefficients
SE <- c(x=.09, y=.09, x2=.07, y2=.07, xy=.11)
# covariances for specific coefficients:
COV <- c(x_y= -.000, x2_y2 = .001, x2_xy = -.003, y2_xy = -.004)
RSA.ST(x = .131, y = .382, x2 = .074, xy = .002, y2 = .039, SE=SE, COV=COV, df=181)

# 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)
```

---

selfother

*A fake data set on self-other agreement*


---

## Description

The data set selfother is a self-generated fake data set which can, for example, be used to try out the inclusion of control variables in the RSA() function and to try out cubic RSA. The variables in the data set are meant to reflect the following constructs:

## Format

A data frame with 800 rows and 9 variables

**Details**

- IQ\_self Self-rated intelligence (on IQ scale)
- IQ\_friend Friend-rated intelligence (on IQ scale)
- harmony Level of harmony in typical interactions between the target person and the friend
- VI\_self Self-reported value importance
- VI\_partner Partner-reported value importance
- distance Emotional distance felt toward the partner
- age Age of the target person
- int Typical number of interactions between the target person and the friend (who provided the intelligence rating) per week
- liking Target person's rating about how much he/she likes the friend

# Index

## \* datasets

- motcon, [13](#)
- motcon2, [14](#)
- selfother, [29](#)

aictab, [2](#)

caRange, [4](#)  
clRange, [5](#)  
colorRampPalette, [20](#)  
compare, [6](#), [12](#), [13](#), [25](#)  
compare2, [6](#)  
confint (confint.RSA), [7](#)  
confint.RSA, [7](#), [24](#), [25](#)

demoRSA, [8](#), [21](#), [25](#)  
demoSRR (demoRSA), [8](#)  
demoSRRR (demoRSA), [8](#)

fitted.RSA, [11](#)

getPar, [11](#)

modeltree, [6](#), [12](#)  
motcon, [13](#)  
motcon2, [14](#)  
movieRSA, [14](#)

plot, [4](#), [5](#)  
plotRSA, [10](#), [15](#), [16](#), [25](#)

resid (residuals.RSA), [22](#)  
residuals.RSA, [22](#)  
RSA, [7](#), [8](#), [10](#), [12](#), [13](#), [21](#), [22](#), [28](#)  
RSA.ST, [25](#), [27](#)

selfother, [29](#)  
sem, [24](#)