Package: apim (via r-universe)

June 4, 2026

Title Actor-Partner Interdependence Model Convenience Functions

Version 0.2.1

Description Utilities for Actor-Partner Interdependence Models (APIM).

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Imports dplyr, ggplot2, glmmTMB, Matrix, parallel, plotly, stats, tidyr

Remotes RobAckerman/glmmTMB/glmmTMB

URL `https://github.com/RobAckerman/apim`

BugReports `https://github.com/RobAckerman/apim/issues`

Depends R (>= 4.1.0)

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev

Repository https://robackerman.r-universe.dev

Date/Publication 2026-06-04 01:08:06 UTC

RemoteUrl https://github.com/RobAckerman/apim

RemoteRef HEAD

RemoteSha cc6912313cd2692f3a69b4021f799f86977d87ab

Contents

`check_rsa_requirements`

Check Preliminary Requirements for Dyadic Response Surface Analysis

---

Description

Evaluates whether the data meet the three requirements necessary for meaningful response surface analysis (RSA) results: (1) adequate variance in both actor and partner scores, (2) sufficient representation of over-estimators and under-estimators (at least `min_pct\` category), and (3) a statistically significant R-squared from the polynomial mixed model. Results are reported for the overall sample and, optionally, separately for each level of a moderator variable.

Usage

```
check_rsa_requirements(
  data,
  outcome,
  actor,
  partner,
  re_term,
  moderator = NULL,
  mod_labels = NULL,
  congruence_threshold = 0.5,
  min_pct = 10
)
```

Arguments

`data`	A data frame.
`outcome`	Character string. Name of the outcome variable.
`actor`	Character string. Name of the (centered) actor predictor.
`partner`	Character string. Name of the (centered) partner predictor.
`re_term`	Character string. The random effects term exactly as it should appear in the model formula, e.g. `"cs(0 + as.factor(ECGender_A) | DyadID)"`. Passed verbatim to `glmmTMB`.

`moderator`	Character string or `NULL`. Name of a grouping variable. When supplied, bias percentages are reported separately for each level. R-squared is always from the full sample. Default `NULL`.
`mod_labels`	Named character vector or `NULL`. Labels for moderator levels, e.g. `c("-1" = "Women", "1" = "Men")`. Default `NULL` uses raw moderator values as labels.
`congruence_threshold`	Numeric. Cutpoint applied to the standardized difference (partner_z - actor_z) for classifying cases as congruent. Default `0.5`, matching `RSA::summary.RSA`.
`min_pct`	Numeric. Minimum percentage of cases required in each bias category. Default `10`.

Details

Participants are classified into three categories based on the standardized difference between actor and partner scores, matching the convention in `RSA::summary.RSA` (Schönbrodt et al., 2018). Both variables are first standardized to a common pooled mean and SD, then the difference (partner_z - actor_z) is computed:

Over-estimator Standardized difference > `congruence_threshold` (partner exceeds actor).

Under-estimator Standardized difference < `-congruence_threshold` (actor exceeds partner).

Congruent |standardized difference| <= `congruence_threshold`.

R-squared and its significance are estimated using `estimate_Rsq`, which fits null and polynomial `glmmTMB` models with the same random effect structure specified in `re_term` and uses a deviance difference test. This properly accounts for dyadic non-independence, unlike the OLS approach used in `RSA::summary.RSA`. R-squared is always estimated on the full sample. When a moderator is supplied, bias percentages are reported separately per group but R-squared is not re-estimated per group.

Value

Invisibly returns a `data.frame` with one row per group.

References

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

Edwards, J. R. (1994). The study of congruence in organizational behavior research: Critique and a proposed alternative. *Organizational Behavior and Human Decision Processes*, 58(1), 51–100.

Examples

```
## Not run:
check_rsa_requirements(
  data        = pairwise_data,
  outcome     = "RelSat_A",
  actor       = "c_PosEmotion_A",
  partner     = "c_PosEmotion_P",
  re_term     = "cs(0 + as.factor(ECGender_A) | DyadID)",
```

```
  moderator  = "ECGender_A",
  mod_labels = c("-1" = "Women", "1" = "Men")
)

## End(Not run)
```

`deviance_diff_test` *Deviance Difference Test for APIM Models*

Description

Computes and prints the chi-square statistic, degrees of freedom, and p-value for a deviance difference test comparing a reduced and full model. Accepts either fitted `gls` or `glmmTMB` model objects directly, or raw deviance values and parameter counts for use with output from other software.

When comparing models that differ in their fixed effects, both models must be estimated using Maximum Likelihood (ML). When comparing models that differ only in their random effects, REML estimation is acceptable.

Usage

```
deviance_diff_test(
  model_r = NULL,
  model_f = NULL,
  deviance_r = NULL,
  deviance_f = NULL,
  p_r = NULL,
  p_f = NULL
)
```

Arguments

`model_r`	A fitted `gls` or `glmmTMB` reduced model object. Required if not supplying raw deviance values.
`model_f`	A fitted `gls` or `glmmTMB` full model object. Required if not supplying raw deviance values.
`deviance_r`	Numeric. Deviance for the reduced model. Only required when not supplying model objects (e.g. when using output from other software).
`deviance_f`	Numeric. Deviance for the full model. Only required when not supplying model objects.
`p_r`	Integer. Number of parameters in the reduced model. Only required when not supplying model objects.
`p_f`	Integer. Number of parameters in the full model. Only required when not supplying model objects.

Value

Invisibly returns a `data.frame` with columns `deviance_r`, `deviance_f`, `p_r`, `p_f`, `chi2`, `df`, and `p`.

Examples

```
## Not run:
# Using model objects
deviance_diff_test(model_r = m_reduced, model_f = m_full)

# Using raw values from other software
deviance_diff_test(deviance_r = 3689.3, deviance_f = 3674.1,
                   p_r = 10, p_f = 12)

## End(Not run)
```

`estimate_Rsq` *Estimate and Test R-squared for APIM Models*

Description

Reports the estimate(s) of R-squared and the corresponding deviance difference test(s) for the indistinguishable and distinguishable Actor-Partner Interdependence Model (APIM).

If researchers use the `nlme` (with the `gls` function) or `glmmTMB` packages, they need only specify the full model object with the `model_full` argument. Either ML or REML estimation can be used for the supplied model; the function handles refitting internally.

By default, the function assumes an indistinguishable APIM. If the model is distinguishable, researchers must specify `indistinguishable = FALSE` and supply the names of the dummy-coded variables for the two dyad members via `person_1` and `person_2`.

For `gls` models with `indistinguishable = FALSE`, the model must be in the two-intercept form (e.g. `RelSat_A ~ 0 + man + woman + man:c_Amity_A + woman:c_Amity_A`). If your model uses the interaction approach, use `switch_to_twoint()` first. The `person_1` argument is assumed to correspond to the reference level of `varIdent` (the level with ratio = 1.0, determined by the first value of the grouping variable encountered in the data) and `person_2` to the non-reference level. A note is printed at runtime showing which factor levels correspond to `person_1` and `person_2` so you can verify the mapping is correct.

For `glmmTMB` models with `indistinguishable = FALSE`, the random effects term must use explicit dummy-coded variables (e.g. `cs(0 + man + woman | DyadID)`) rather than `as.factor()`. The names supplied to `person_1` and `person_2` must match the dummy variable names exactly.

To ensure valid estimates of R-squared and deviance difference tests, the supplied model object should not contain any missing data for the predictors.

If researchers use other software to estimate the models, they must provide all individual quantities needed to estimate and test R-squared. When supplying these values, researchers must ensure that the deviances are from models estimated with ML and that the residual variances are from models estimated with REML.

Usage

```
estimate_Rsq(
  model_full = NULL,
  indistinguishable = TRUE,
  deviance_null = NULL,
  deviance_full = NULL,
  deviance_p1 = NULL,
  deviance_p2 = NULL,
  p_null = NULL,
  p_full = NULL,
  p_p1 = NULL,
  p_p2 = NULL,
  resvar_null = NULL,
  resvar_full = NULL,
  resvar_null_p1 = NULL,
  resvar_null_p2 = NULL,
  resvar_full_p1 = NULL,
  resvar_full_p2 = NULL,
  person_1 = "Person 1",
  person_2 = "Person 2"
)
```

Arguments

`model_full`	A fitted `gls` or `glmmTMB` full model object.
`indistinguishable`	Logical. If `TRUE` (default), the function assumes an indistinguishable APIM. If `FALSE`, a distinguishable APIM is assumed.
`deviance_null`	Numeric. ML deviance for the null model. Only required when not supplying model objects.
`deviance_full`	Numeric. ML deviance for the full model. Only required when not supplying model objects.
`deviance_p1`	Numeric. ML deviance for the person 1 model. Only required for the distinguishable case when not supplying model objects.
`deviance_p2`	Numeric. ML deviance for the person 2 model. Only required for the distinguishable case when not supplying model objects.
`p_null`	Integer. Number of parameters in the null model. Only required when not supplying model objects.
`p_full`	Integer. Number of parameters in the full model. Only required when not supplying model objects.
`p_p1`	Integer. Number of parameters in the person 1 model. Only required for the distinguishable case when not supplying model objects.
`p_p2`	Integer. Number of parameters in the person 2 model. Only required for the distinguishable case when not supplying model objects.
`resvar_null`	Numeric. REML residual variance from the null model. Only required for the indistinguishable case when not supplying model objects.

`resvar_full`	Numeric. REML residual variance from the full model. Only required for the indistinguishable case when not supplying model objects.
`resvar_null_p1`	Numeric. REML residual variance for person 1 from the null model. Only required for the distinguishable case when not supplying model objects.
`resvar_null_p2`	Numeric. REML residual variance for person 2 from the null model. Only required for the distinguishable case when not supplying model objects.
`resvar_full_p1`	Numeric. REML residual variance for person 1 from the full model. Only required for the distinguishable case when not supplying model objects.
`resvar_full_p2`	Numeric. REML residual variance for person 2 from the full model. Only required for the distinguishable case when not supplying model objects.
`person_1`	Character. Display label and variable name for the first dyad member. For `glmmTMB` models, must match the dummy-coded variable name in the random effects term exactly (e.g. `"man"`). For `gls` models, assumed to correspond to the reference level of `varIdent` (ratio = 1.0). Defaults to `"Person 1"`.
`person_2`	Character. Display label and variable name for the second dyad member. For `glmmTMB` models, must match the dummy-coded variable name exactly. For `gls` models, assumed to correspond to the non-reference level of `varIdent`. Defaults to `"Person 2"`.

Value

Invisibly returns a `data.frame` with columns `person`, `R2`, `chi2`, `df`, and `p`. For the indistinguishable case the data frame has one row; for the distinguishable case it has three rows (omnibus, person 1, person 2).

Examples

```
## Not run:
# Indistinguishable APIM - supply model object
estimate_Rsq(model_full = fullmodel_REML)

# Indistinguishable APIM - supply raw values
estimate_Rsq(deviance_null = 1472.455, deviance_full = 1341.497,
             p_null = 3, p_full = 5,
             resvar_null = 0.722, resvar_full = 0.562)

# Distinguishable APIM - gls, must be in two-intercept form
m_twoint <- switch_to_twoint(m_interact,
                             disting_variable = "ECGender_A",
                             disting_level_1  = "man",
                             disting_level_2  = "woman")
estimate_Rsq(m_twoint,
             indistinguishable = FALSE,
             person_1 = "man", person_2 = "woman")

# Distinguishable APIM - glmmTMB, RE must use explicit dummy codes
estimate_Rsq(model_full = fulldismodel_REML,
             indistinguishable = FALSE,
             person_1 = "man", person_2 = "woman")
```

```
# Distinguishable APIM - supply raw values
estimate_Rsq(indistinguishable = FALSE,
             deviance_null = 1187.010, deviance_full = 1137.756,
             deviance_p1 = 1176.509, deviance_p2 = 1169.441,
             p_null = 5, p_full = 9, p_p1 = 7, p_p2 = 7,
             resvar_null_p1 = 0.429, resvar_null_p2 = 0.534,
             resvar_full_p1 = 0.389, resvar_full_p2 = 0.473,
             person_1 = "man", person_2 = "woman")

## End(Not run)
```

`johnson_neyman` *Johnson-Neyman Regions of Significance for Two-Way Interactions*

Description

Computes and plots Johnson-Neyman regions of significance for the effect of a focal predictor across the observed range of a moderator variable. Supports models fitted with `glmmTMB`, `lme`, or `gls`. Three methods are available for degrees of freedom via `df_method`:

- `"z"` **(default)** Large-sample z-test for all model types.
- `"t"` t-test using residual df from the model. Available for `gls` and `lme` only. For `lme`, residual df are used rather than per-term df. Not available for `glmmTMB`; falls back to `"z"` with a warning.
- `"satterthwaite"` Satterthwaite df computed on a coarse grid and interpolated across the moderator range. Available for `glmmTMB` with `dispformula = ~0` only. Falls back to `"t"` for `gls` and `lme` with a warning.

Usage

```
johnson_neyman(
  model,
  pred,
  modx,
  data,
  alpha = 0.05,
  n_mod = 1000,
  df_method = c("z", "t", "satterthwaite"),
  sw_grid = 20,
  eps = 0.001,
  n_cores = parallel::detectCores(logical = FALSE),
  title = NULL,
  ylabel = NULL,
  verbose = FALSE
)
```

Arguments

`model`	A fitted model object of class `glmmTMB`, `lme`, or `gls` containing an interaction between `pred` and `modx`.
`pred`	Character string. Name of the focal predictor.
`modx`	Character string. Name of the moderator variable.
`data`	A `data.frame` containing the variables used to fit `model`.
`alpha`	Numeric. Significance level. Default is `0.05`.
`n_mod`	Integer. Number of points in the moderator grid. Default is `1000`.
`df_method`	Character string. One of `"z"` (default), `"t"`, or `"satterthwaite"`. See Description.
`sw_grid`	Integer. Number of coarse grid points for Satterthwaite df computation when `df_method = "satterthwaite"`. Df values are interpolated to the full `n_mod`-point grid. Default is `20`.
`eps`	Numeric. Step size for Jacobian in Satterthwaite computation. Default is `1e-3`.
`n_cores`	Integer. Number of parallel workers for Satterthwaite computation. Default is all physical cores.
`title`	Character string. Plot title. Auto-generated if `NULL`.
`ylabel`	Character string. Y-axis label. Auto-generated if `NULL`.
`verbose`	Logical. Print progress during Satterthwaite computation. Default is `FALSE`.

Value

Invisibly returns a `data.frame` with columns:

moderator Moderator values across the grid.

simple_slope Simple slope of `pred`.

se_slope Standard error of the simple slope.

df Degrees of freedom.

p_value Two-tailed p-value.

Examples

```
## Not run:
# glmmTMB with Satterthwaite df
johnson_neyman(
  model     = ind_moderation_socA,
  pred      = "c_PosBehavior_A",
  modx      = "c_Support_A",
  data      = pairwise_indisting,
  df_method = "satterthwaite",
  sw_grid   = 20
)

# gls with residual df t-test
johnson_neyman(
```

```
  model     = ind_moderation_bdyad_gls,
  pred      = "c_PosBehavior_A",
  modx      = "c_Rellengthyrs",
  data      = pairwise_indisting,
  df_method = "t"
)

# lme with residual df t-test
johnson_neyman(
  model     = ind_moderation_bdyad_lme,
  pred      = "c_PosBehavior_A",
  modx      = "c_Rellengthyrs",
  data      = pairwise_indisting,
  df_method = "t"
)

## End(Not run)
```

`kr_glmmTMB_no_resid` *Kenward-Roger df for glmmTMB models with dispformula = ~0*

Description

Works for any random-effect covariance structure (us, homcs, ar1, toep, ...) by computing dV/dtheta numerically via model refits – no tape interaction.

Usage

```
kr_glmmTMB_no_resid(
  model,
  data = NULL,
  eps = 1e-04,
  n_cores = parallel::detectCores(logical = FALSE),
  verbose = TRUE
)
```

Arguments

`model`	fitted glmmTMB object with dispformula = ~0 and REML = TRUE
`eps`	step size for central differences (default 1e-4)
`n_cores`	parallel workers for dV/dtheta (default: all physical cores)
`verbose`	print progress and results

Value

data.frame: Term, Estimate, SE (KR-adjusted), df, t, p attribute "vcov_kr": KR-adjusted fixed-effect vcov matrix

---

`linear_contrast` *Test a Linear Contrast of Fixed Effects*

Description

Tests an arbitrary linear combination of fixed effects $L'\beta$ for models fitted with `glmmTMB`, `gls`, or `lme`. Three methods are available for computing degrees of freedom, controlled by the `df_method` argument:

- `"z"` **(default)** Large-sample z-test for all model types.
- `"t"` t-test using residual df from the model. Available for `gls` and `lme` only. For `lme`, residual df are used for all terms rather than per-term df. Not available for `glmmTMB`; falls back to `"z"` with a warning.
- `"satterthwaite"` Satterthwaite df via numerical differentiation of the contrast variance. Available for `glmmTMB` with `dispformula = ~0` only. Falls back to `"t"` for `gls` and `lme` with a warning.

Usage

```
linear_contrast(
  model,
  L,
  label = "Contrast",
  alpha = 0.05,
  df_method = c("z", "t", "satterthwaite"),
  eps = 0.001,
  n_cores = parallel::detectCores(logical = FALSE),
  verbose = FALSE
)
```

Arguments

`model`	A fitted model object of class `glmmTMB`, `gls`, or `lme`.
`L`	A named numeric vector of contrast weights. Names must match coefficient names in the model. Coefficients not named in `L` are assumed to have weight zero.
`label`	A character string label for the contrast. Default is `"Contrast"`.
`alpha`	Numeric. Significance level for confidence intervals. Default is `0.05`.
`df_method`	Character string specifying the method for degrees of freedom. One of `"z"` (default), `"t"`, or `"satterthwaite"`. See Details.
`eps`	Numeric. Step size for Jacobian in Satterthwaite computation. Only used when `df_method = "satterthwaite"` and model is `glmmTMB` with `dispformula = ~0`. Default is `1e-3`.
`n_cores`	Integer. Number of parallel workers for Satterthwaite computation. Default is all physical cores.
`verbose`	Logical. Print progress during Satterthwaite computation. Default is `FALSE`.

Value

Invisibly returns a `data.frame` with columns:

label Contrast label.

estimate Estimated value of $L'\beta$.

se Standard error.

df Degrees of freedom.

t t- or z-statistic.

p Two-tailed p-value.

ci_lower Lower confidence interval bound.

ci_upper Upper confidence interval bound.

Examples

```
## Not run:
# simple slope of c_PosBehavior_A at low support (w = -1.31)
# glmmTMB with Satterthwaite df
linear_contrast(
  model     = ind_moderation_socA,
  L         = c(c_PosBehavior_A = 1,
                "c_PosBehavior_A:c_Support_A" = -1.31),
  label     = "Actor simple slope at Low Support",
  df_method = "satterthwaite"
)

# gls with residual df t-test
linear_contrast(
  model     = ind_moderation_bdyad_gls,
  L         = c(c_PosBehavior_A = 1,
                "c_PosBehavior_A:c_Rellengthyrs" = -6.04),
  label     = "Actor simple slope at Low Rellengthyrs",
  df_method = "t"
)

# lme with residual df t-test
linear_contrast(
  model     = ind_moderation_bdyad_lme,
  L         = c(c_PosBehavior_A = 1,
                "c_PosBehavior_A:c_Rellengthyrs" = -6.04),
  label     = "Actor simple slope at Low Rellengthyrs",
  df_method = "t"
)

## End(Not run)
```

`pick_a_point` *Pick-a-Point Analysis for Two-Way Interactions*

Description

Computes simple slopes of a focal predictor at specified values of a moderator variable (the pick-a-point or spotlight approach). Supports models fitted with `glmmTMB`, `lme`, or `gls`. Three methods are available for degrees of freedom via `df_method`:

`"z"` **(default)** Large-sample z-test for all model types.

`"t"` t-test using residual df. Available for `gls` and `lme` only. For `lme`, residual df are used for all simple slopes rather than per-term df. Not available for `glmmTMB`; falls back to `"z"` with a warning.

`"satterthwaite"` Satterthwaite df computed separately for each simple slope via numerical differentiation of the contrast variance. Available for `glmmTMB` with `dispformula = ~0` only. Falls back to `"t"` for `gls` and `lme` with a warning.

Usage

```
pick_a_point(
  model,
  pred,
  modx,
  data,
  points = c("mean-sd", "mean", "mean+sd"),
  alpha = 0.05,
  df_method = c("z", "t", "satterthwaite"),
  eps = 0.001,
  n_cores = parallel::detectCores(logical = FALSE),
  verbose = FALSE
)
```

Arguments

`model`	A fitted model object of class `glmmTMB`, `lme`, or `gls` containing an interaction between `pred` and `modx`.
`pred`	Character string. Name of the focal predictor.
`modx`	Character string. Name of the moderator variable.
`data`	A `data.frame` containing the variables used to fit `model`.
`points`	Either a character vector of keywords (`"mean-sd"`, `"mean"`, `"mean+sd"`) or a numeric vector of specific moderator values. Default is `c("mean-sd", "mean", "mean+sd")`.
`alpha`	Numeric. Significance level. Default is `0.05`.
`df_method`	Character string. One of `"z"` (default), `"t"`, or `"satterthwaite"`. See Description.

`eps`	Numeric. Step size for Jacobian in Satterthwaite computation. Default is `1e-3`.
`n_cores`	Integer. Number of parallel workers for Satterthwaite computation. Default is all physical cores.
`verbose`	Logical. Print progress during Satterthwaite computation. Default is `FALSE`.

Value

Invisibly returns a `data.frame` with columns:

Label Description of the moderator value.

Moderator_Value Numeric value of the moderator.

Simple_Slope Estimated simple slope of `pred`.

SE Standard error of the simple slope.

df Degrees of freedom.

t t- or z-statistic.

p Two-tailed p-value.

CI_lower Lower confidence interval bound.

CI_upper Upper confidence interval bound.

Sig Significance indicator (`*` if `p` < `alpha`).

Examples

```
## Not run:
# glmmTMB with Satterthwaite df
pick_a_point(
  model     = ind_moderation_socA,
  pred      = "c_PosBehavior_A",
  modx      = "c_Support_A",
  data      = pairwise_indisting,
  df_method = "satterthwaite"
)

# gls with residual df t-test
pick_a_point(
  model     = ind_moderation_bdyad_gls,
  pred      = "c_PosBehavior_A",
  modx      = "c_Rellengthyrs",
  data      = pairwise_indisting,
  df_method = "t"
)

# lme with residual df t-test
pick_a_point(
  model     = ind_moderation_bdyad_lme,
  pred      = "c_PosBehavior_A",
  modx      = "c_Rellengthyrs",
  data      = pairwise_indisting,
  df_method = "t"
```

```
)

## End(Not run)
```

`plot_dyadic_rsa` *Plot Dyadic Response Surface Analysis Surfaces*

Description

Produces response surface plots from the output of `run_dyadic_rsa`. Two plot types are available: an interactive 3D surface using `plotly` (default), and a static black-and-white wireframe plot using `lattice` matching the style of `RSA::plotRSA`. One panel is produced per surface (overall, and one per group in `mod_values`). The actor axis is reversed by default to match `RSA::plotRSA` orientation. The line of congruence (actor = partner) and line of incongruence (actor = -partner) are overlaid on each surface.

Usage

```
plot_dyadic_rsa(
  rsa_out,
  actor,
  partner,
  moderator = NULL,
  mod_values = NULL,
  type = "interactive",
  xlim = c(-3, 3),
  ylim = c(-3, 3),
  n = NULL,
  xlab = "Actor",
  ylab = "Partner",
  zlab = "Outcome",
  rotation = list(x = -63, y = 32, z = 15)
)
```

Arguments

`rsa_out`	A list returned by `run_dyadic_rsa`.
`actor`	Character string. Name of the actor predictor — must match the value passed to `run_dyadic_rsa`.
`partner`	Character string. Name of the partner predictor — must match the value passed to `run_dyadic_rsa`.
`moderator`	Character string or `NULL`. Name of the moderator variable. Default `NULL`.
`mod_values`	Named list or `NULL`. Moderator values for group-specific surfaces. Default `NULL`.
`type`	Character string. Plot type: `"interactive"` for plotly (default) or `"static"` for lattice wireframe.

`xlim`	Numeric vector of length 2. Range of the actor axis. Default `c(-3, 3)`. Note: the actor axis is displayed reversed (high values at front-left) to match `RSA::plotRSA` orientation.
`ylim`	Numeric vector of length 2. Range of the partner axis. Default `c(-3, 3)`.
`n`	Integer. Number of grid points along each axis. Default `60` for interactive, `21` for static.
`xlab`	Character string. Label for the actor axis. Default `"Actor"`.
`ylab`	Character string. Label for the partner axis. Default `"Partner"`.
`zlab`	Character string. Label for the outcome axis. Default `"Outcome"`.
`rotation`	Named list. Rotation angles for static plot. Default `list(x = -63, y = 32, z = 15)` matching `RSA::plotRSA`.

Value

Invisibly returns a named list of plot objects, one per panel.

References

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

Examples

```
## Not run:
rsa_out <- run_dyadic_rsa(
  data       = pairwise_data,
  outcome    = "RelSat_A",
  actor      = "c_PosEmotion_A",
  partner    = "c_PosEmotion_P",
  re_term    = "cs(0 + as.factor(ECGender_A) | DyadID)",
  moderator  = "ECGender_A",
  mod_values = list(Men = 1, Women = -1)
)

# Interactive plotly
plot_dyadic_rsa(
  rsa_out    = rsa_out,
  actor      = "c_PosEmotion_A",
  partner    = "c_PosEmotion_P",
  moderator  = "ECGender_A",
  mod_values = list(Men = 1, Women = -1),
  type       = "interactive"
)

# Static B&W wireframe
plot_dyadic_rsa(
  rsa_out    = rsa_out,
  actor      = "c_PosEmotion_A",
  partner    = "c_PosEmotion_P",
  moderator  = "ECGender_A",
```

```
  mod_values = list(Men = 1, Women = -1),
  type       = "static"
)

## End(Not run)
```

---

run_dyadic_rsa *Fit a Dyadic Response Surface Analysis Model*

Description

Fits a dyadic response surface analysis (RSA) model using `glmmTMB`, computes RSA parameters a1–a5 for the overall surface and optionally for subgroups defined by a moderator variable, and tests moderation of the response surface. Validated against the dyadic RSA framework of Schönbrodt, Humberg, & Nestler (2018).

Usage

```
run_dyadic_rsa(
  data,
  outcome,
  actor,
  partner,
  re_term,
  moderator = NULL,
  mod_values = NULL,
  alpha = 0.05,
  df_method = "z",
  reml = TRUE
)
```

Arguments

`data`	A data frame.
`outcome`	Character string. Name of the outcome variable.
`actor`	Character string. Name of the (centered) actor predictor.
`partner`	Character string. Name of the (centered) partner predictor.
`re_term`	Character string. The random effects term exactly as it should appear in the model formula, e.g. `"cs(0 + as.factor(Gender) | DyadID)"`. Passed verbatim.
`moderator`	Character string or `NULL`. Name of the moderator variable (e.g. a distinguishing variable such as gender). When supplied, all five RSA terms are interacted with the moderator. Default `NULL`.

`mod_values`	Named list or `NULL`. Values of the moderator at which to estimate group-specific surfaces, e.g. `list(Men = 1, Women = -1)`. Required when `moderator` is supplied and moderation tests are desired. Exactly two values triggers moderation tests. Default `NULL`.
`alpha`	Numeric. Significance level for confidence intervals. Default `0.05`.
`df_method`	Character string. Degrees-of-freedom method passed to `linear_contrast`. One of `"z"` (default), `"t"`, or `"satterthwaite"`.
`reml`	Logical. Whether to use REML estimation. Default `TRUE`, which is preferred for variance component estimation. Set to `FALSE` to match maximum likelihood estimators (e.g. for validation against lavaan or for likelihood ratio tests).

Details

The function automatically constructs the five polynomial terms required for RSA (actor^2, partner^2, actor×partner) and builds the fixed-effects formula. When a moderator is supplied, all five RSA terms are interacted with the moderator, and surfaces are estimated at each value supplied in `mod_values` as well as at the grand mean (moderator = 0), which under effect coding equals the unweighted average of the group surfaces.

RSA parameters are estimated as linear contrasts of the fixed effects using `linear_contrast`:

a1 Slope along the line of congruence (actor + partner).

a2 Curvature along the line of congruence (actor^2 + actor×partner + partner^2).

a3 Slope along the line of incongruence (actor - partner).

a4 Curvature along the line of incongruence (actor^2 - actor×partner + partner^2).

a5 Rotation of the principal axis (actor^2 - partner^2).

Results have been verified to be numerically identical (differences $< 1e-5$) to the lavaan-based dyadic RSA implementation of Schönbrodt et al. (2018) when both are estimated with maximum likelihood.

Value

Invisibly returns a named list with components:

model The fitted `glmmTMB` model object.

results A named list of `data.frames` with RSA contrast results. Always includes `$overall`. When `mod_values` is supplied, also includes one element per group and `$moderation`.

coefs Named numeric vector of fixed-effect estimates.

nms Character vector of fixed-effect names.

References

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

Examples

```
## Not run:
# Without moderator — single overall surface
rsa_out <- run_dyadic_rsa(
  data      = pairwise_data,
  outcome   = "RelSat_A",
  actor     = "c_PosEmotion_A",
  partner   = "c_PosEmotion_P",
  re_term   = "cs(0 + as.factor(ECGender_A) | DyadID)",
  df_method = "z"
)

# With gender moderator — overall, group, and moderation surfaces
rsa_out <- run_dyadic_rsa(
  data       = pairwise_data,
  outcome    = "RelSat_A",
  actor      = "c_PosEmotion_A",
  partner    = "c_PosEmotion_P",
  re_term    = "cs(0 + as.factor(ECGender_A) | DyadID)",
  moderator  = "ECGender_A",
  mod_values = list(Men = 1, Women = -1),
  df_method  = "z"
)

# Access results
rsa_out$model                 # glmmTMB model object
rsa_out$results$overall       # overall surface
rsa_out$results$Men           # men's surface
rsa_out$results$Women         # women's surface
rsa_out$results$moderation    # moderation tests

## End(Not run)
```

`summary_glmmTMB_kr` *Kenward-Roger summary for glmmTMB models*

Description

Kenward-Roger summary for glmmTMB models

Usage

```
summary_glmmTMB_kr(
  model,
  eps = 1e-04,
  n_cores = 1L,
  digits = 5,
  alpha = 0.05,
```

```
  verbose = FALSE
)
```

Arguments

`model`	a fitted glmmTMB object (REML recommended)
`eps`	step size for central differences in dV/dtheta (only used when dispformula = ~0)
`n_cores`	parallel workers for dV/dtheta refits (only used when dispformula = ~0)
`digits`	decimal places in printed output (currently unused, kept for API consistency with satterthwaite version)
`alpha`	significance level for confidence intervals
`verbose`	print progress from kr_glmmTMB_no_resid()

Value

invisibly: list(kenward_roger, vc, fit_stats) kenward_roger: data.frame with Term, Estimate, SE, df, t, p, lower, upper vc: VarCorr(model)$cond fit_stats: named numeric vector of AIC, BIC, logLik, -2*log(L), df.resid

---

`summary_glmmTMB_satterthwaite`

Satterthwaite Summary for glmmTMB Models

Description

Produces a formatted summary for a fitted `glmmTMB` model using Satterthwaite degrees of freedom for fixed effects. Confidence intervals are computed using Satterthwaite df and included in the fixed effects table.

Usage

```
summary_glmmTMB_satterthwaite(
  model,
  eps = 0.001,
  digits = 5,
  alpha = 0.05,
  verbose = FALSE
)
```

Arguments

`model`	A fitted `glmmTMB` model object.
`eps`	Numeric. Step size for central differences in the Jacobian. Only used when `dispformula = ~0`. Default is `1e-3`.
`digits`	Integer. Number of digits for rounding. Default is `5`.

`alpha`	Numeric. Significance level for confidence intervals. Default is `0.05`, giving 95% confidence intervals.
`verbose`	Logical. If `TRUE`, prints progress messages. Default is `FALSE`.

Value

Invisibly returns a list with elements: satterthwaite (data frame with Term, Estimate, SE, df, t, p, lower, upper), vc (VarCorr output), and fit_stats (named vector of AIC, BIC, logLik, -2*log(L), df.resid).

Examples

```
## Not run:
m <- glmmTMB::glmmTMB(y ~ x, data = dat)
summary_glmmTMB_satterthwaite(m)
summary_glmmTMB_satterthwaite(m, alpha = 0.10)

## End(Not run)
```

`switch_to_interact` *Switch a Distinguishable APIM model with two-intercept approach to the interaction approach*

Description

Reformulates a fitted `gls` or `glmmTMB` Distinguishable Actor-Partner Interdependence Model (APIM) from the two-intercept parameterisation to the equivalent interaction approach. The two parameterisations are statistically equivalent; a numerical ML deviance check confirms this before returning.

For `gls` models, non-formula arguments (`correlation`, `weights`, `na.action`) are preserved automatically by `update()`.

Usage

```
switch_to_interact(object = NULL, disting_variable = NULL)
```

Arguments

`object`	A fitted `gls` or `glmmTMB` model object specified using the two-intercept approach.
`disting_variable`	Character. Name of the effect-coded distinguishing variable (e.g. `"ECGender_A"`).

Value

A fitted model object of the same class as `object`, reformulated using the interaction approach and estimated with the same method as the original (REML for `glmmTMB`; the original `method` for `gls`). Stops with an informative message if the ML deviance of the reformulated model differs from the original.

Examples

```
## Not run:
# gls with dummy codes
m_twoint <- gls(RelSat_A ~ 0 + man + woman +
                            man:c_Amity_A + woman:c_Amity_A +
                            man:c_Amity_P + woman:c_Amity_P,
                correlation = corCompSymm(form = ~1 | DyadID),
                weights     = varIdent(form = ~1 | ECGender_A),
                data        = pairwise_disting)

m_interact <- switch_to_interact(object           = m_twoint,
                                 disting_variable = "ECGender_A")

# gls with as.factor() distinguishing variable
m_twoint_f <- gls(RelSat_A ~ 0 + as.factor(ECGender_A) +
                              as.factor(ECGender_A):c_Amity_A +
                              as.factor(ECGender_A):c_Amity_P,
                  correlation = corCompSymm(form = ~1 | DyadID),
                  weights     = varIdent(form = ~1 | ECGender_A),
                  data        = pairwise_disting)

m_interact_f <- switch_to_interact(object           = m_twoint_f,
                                   disting_variable = "ECGender_A")

# glmmTMB
m_twoint_tmb <- glmmTMB(RelSat_A ~ 0 + man + woman +
                                    man:c_Amity_A + woman:c_Amity_A +
                                    man:c_Amity_P + woman:c_Amity_P +
                                    cs(0 + man + woman | DyadID),
                         dispformula = ~0, REML = TRUE,
                         data = pairwise_disting)

m_interact_tmb <- switch_to_interact(object           = m_twoint_tmb,
                                     disting_variable = "ECGender_A")

## End(Not run)
```

`switch_to_twoint` *Switch a Distinguishable APIM model with interaction approach to the two-intercept approach*

Description

Reformulates a fitted `gls` or `glmmTMB` Distinguishable Actor-Partner Interdependence Model (APIM) from the interaction approach parameterisation to the equivalent two-intercept approach. The two parameterisations are statistically equivalent; a numerical ML deviance check confirms this before returning.

The two-intercept approach suppresses the global intercept and estimates a separate intercept for each dyad member, with all predictors crossed with their respective group dummy variable using `:`.

For `gls` models, non-formula arguments (`correlation`, `weights`, `na.action`) are preserved automatically by `update()`.

Usage

```
switch_to_twoint(
  object = NULL,
  disting_variable = NULL,
  disting_level_1 = NULL,
  disting_level_2 = NULL
)
```

Arguments

`object`	A fitted `gls` or `glmmTMB` model object specified using the interaction approach.
`disting_variable`	Character. Name of the effect-coded distinguishing variable (e.g. `"ECGender_A"`).
`disting_level_1`	Character. Name of the dummy-coded variable for the first dyad member in the dataset (e.g. `"man"`).
`disting_level_2`	Character. Name of the dummy-coded variable for the second dyad member in the dataset (e.g. `"woman"`).

Value

A fitted model object of the same class as `object`, reformulated using the two-intercept approach and estimated with the same method as the original (REML for `glmmTMB`; the original `method` for `gls`). Stops with an informative message if the ML deviance of the reformulated model differs from the original.

Examples

```
## Not run:
# gls
m_interact <- gls(RelSat_A ~ c_Amity_A * ECGender_A +
                              c_Amity_P * ECGender_A,
                  correlation = corCompSymm(form = ~1 | DyadID),
                  weights     = varIdent(form = ~1 | ECGender_A),
                  data        = pairwise_disting)

m_twoint <- switch_to_twoint(object           = m_interact,
                             disting_variable  = "ECGender_A",
                             disting_level_1   = "man",
                             disting_level_2   = "woman")

# glmmTMB
m_interact_tmb <- glmmTMB(RelSat_A ~ c_Amity_A * ECGender_A +
                                      c_Amity_P * ECGender_A +
                                      cs(0 + man + woman | DyadID),
```

```
                           dispformula = ~0, REML = TRUE,
                           data = pairwise_disting)

m_twoint_tmb <- switch_to_twoint(object           = m_interact_tmb,
                                 disting_variable  = "ECGender_A",
                                 disting_level_1   = "man",
                                 disting_level_2   = "woman")

## End(Not run)
```

Index