Package: roundRobinR (via r-universe)

November 16, 2026

Title Manipulate and Analyze Round Robin Dyadic Data

Version 2.0.0

Description Provides utilities for processing and analyzing dyadic data collected using a round-robin design, in which each person in a group rates or interacts with every other person on at least one variable. Data manipulation functions prepare datasets for dyadic data analysis by creating the actor and partner dummy variables required by the social relations model (SRM). Analysis functions implement the SRM using multilevel modeling via a custom 'nlme' covariance class ('pdSRM'), following the approach of Snijders and Kenny (1999) <doi:10.1111/j.1475-6811.1999.tb00204.x> and Knight and Humphrey (2019) <doi:10.1037/0000115-019>. The package estimates group, actor, partner, and relationship variance components along with generalized and dyadic reciprocity correlations, and supports both null and fixed-effects models.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Depends R (>= 3.5.0)

Imports data.table (>= 1.14.0), Matrix, nlme (>= 3.1-150), stats

Suggests numDeriv, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/andrewpknight/roundRobinR

BugReports https://github.com/andrewpknight/roundRobinR/issues

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-09-17 19:12:34 UTC

RemoteUrl https://github.com/andrewpknight/roundrobinr

RemoteRef HEAD

RemoteSha 3d97b2e6929e6a7e3dbd2d571e3d9468916e2f4c

Contents

coef.pdSRM . . . 2
corMatrix.pdSRM . . . 3
cpsrm . . . 4
cpsrm_run . . . 6
cpsrmRun . . . 9
create_cp_dummies . . . 10
create_dummies . . . 12
createDummies . . . 13
pdConstruct.pdSRM . . . 14
pdMatrix.pdSRM . . . 15
pdSRM . . . 16
print.cpsrm . . . 17
sampleDyadData . . . 18
srm_pseudo_rsq . . . 19
srm_run . . . 20
srm_var_pct . . . 21
srmPseudoRSq . . . 22
srmRun . . . 23
srmVarPct . . . 23
summary.cpsrm . . . 24
summary.pdSRM . . . 24

Index **26**

`coef.pdSRM` *Extract Coefficients from a pdSRM Object*

Description

Internal method called by `coef.pdMat` to extract the three underlying SRM parameters: actor standard deviation, partner standard deviation, and actor-partner correlation.

Usage

```
## S3 method for class 'pdSRM'
coef(object, unconstrained = TRUE, ...)
```

Arguments

object	an object inheriting from pdSRM
unconstrained	logical; if TRUE (default) the unconstrained parameterization is returned; if FALSE the three named SRM parameters are returned
...	additional arguments (currently unused)

Value

a named numeric vector with elements std. dev-a, std. dev-p, and corr.

Examples

```
d <- createDummies(
  group.id = "groupId", act.id = "actId", part.id = "partId",
  d = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge.original = TRUE
)
o <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
```

---

corMatrix.pdSRM *Extract Correlation Matrix from a pdSRM Object*

Description

Internal method called by corMatrix to reconstruct the correlation matrix from the compressed SRM parameters.

Usage

```
## S3 method for class 'pdSRM'
corMatrix(object, ...)
```

Arguments

object	an object inheriting from pdSRM
...	additional arguments (currently unused)

Value

the correlation matrix corresponding to the positive-definite matrix represented by `object`, with a `"stdDev"` attribute giving the standard deviations

Examples

```
d <- createDummies(
  group.id = "groupId", act.id = "actId", part.id = "partId",
  d = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge.original = TRUE
)
o <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
```

`cpsrm` *Fit the Co-Partner Social Relations Model from Raw Long-Format Data*

Description

`cpsrm()` is a friendly, tidyverse-style entry point for the Co-Partner SRM. Give it raw long-format data and the names of the outcome, actor, and group columns; it builds the actor/partner dummy matrices with `create_cp_dummies` and fits the model with `cpsrm_run`. Use `cpsrm_run` directly when you need to supply your own dummy matrices (e.g. dummies built once and reused across several model calls).

Usage

```
cpsrm(
  data,
  dv,
  actor_id,
  group_id,
  location_id = NULL,
  weight_partners = TRUE,
  ...
)
```

Arguments

data	a data.frame in long format, one row per person-group observation
dv	character; column name of the outcome variable
actor_id	character; column name of the actor/person identifier
group_id	character; column name of the group identifier
location_id	character or NULL; column name of a higher-level clustering variable (e.g. course, site), if any
weight_partners	logical; passed to both create_cp_dummies (how the dummies are built) and cpsrm_run (how σ^2_P is interpreted) — see create_cp_dummies for details. Default TRUE
...	additional arguments passed through to cpsrm_run (e.g. zero_rho, fixed_effects, method, optimizer, se_method, start, verbose)

Value

an object of class "cpsrm"; see cpsrm_run for the full return value. The dummy-construction output from create_cp_dummies is attached as $dummies for reference (e.g. to inspect $dummies$group_size_table).

See Also

cpsrm_run for the full-control interface, create_cp_dummies for the dummy-matrix construction this wrapper calls internally

Examples

```
# cpsrm() expects one row per PERSON per GROUP, with every other group
# member treated as a simultaneous co-partner (e.g. one row per player
# per team, as in three-person golf teams). sampleDyadData is a
# round-robin/directed-dyad dataset (multiple rows per actor per group,
# one per rated partner) and is NOT the right shape for this function --
# see create_dummies()/srm_run() for that design instead. This example
# simulates a small, correctly-shaped dataset: 30 three-person teams
# drawn from a pool of 90 players.
set.seed(1)
team_dat <- do.call(rbind, lapply(1:30, function(g) {
  data.frame(player = sample(1:90, 3), team = g)
}))
team_dat$score <- rnorm(nrow(team_dat), mean = 70, sd = 3)

fit <- cpsrm(
  data         = team_dat,
  dv           = "score",
  actor_id     = "player",
  group_id     = "team",
  zero_rho     = TRUE,     # keep the toy example fast/well-behaved
  se_method    = "none",
  stage1_maxit = 100,
  stage2_maxit = 100
```

```
)
print(fit)
```

`cpsrm_run` *Run the Co-Partner Social Relations Model*

Description

Fits the Co-Partner Social Relations Model (CP-SRM) using restricted maximum likelihood (REML). The `"block"` method uses the Woodbury matrix identity for fast computation on large datasets; the `"loop"` method constructs the full covariance matrix by iterating over groups.

Usage

```
cpsrm_run(
  dv,
  actor_id,
  group_id,
  data,
  actor_dummies,
  partner_dummies,
  group_sizes = NULL,
  location_id = NULL,
  method = c("block", "loop"),
  zero_rho = FALSE,
  zero_actor = FALSE,
  zero_partner = FALSE,
  zero_group = FALSE,
  zero_location = FALSE,
  weight_partners = TRUE,
  fixed_effects = NULL,
  start = NULL,
  maxit = 200000L,
  stage1_maxit = 1000L,
  stage2_maxit = 1000L,
  tol = 1e-10,
  se_method = c("hessian", "none"),
  rho_boundary = 0.99,
  optimizer = c("Nelder-Mead", "L-BFGS-B"),
  verbose = FALSE,
  verbose_every = 50L
)
```

Arguments

`dv`	character; name of the dependent variable column

actor_id	character; name of the actor identifier column
group_id	character; name of the group identifier column
data	a data.frame containing all required variables
actor_dummies	character vector; actor dummy column names
partner_dummies	character vector; partner dummy column names
group_sizes	integer vector or NULL; per-row group sizes. Pass the group_sizes element from create_cp_dummies
location_id	character or NULL; optional higher-level clustering variable. Default NULL
method	character; "block" (default) or "loop"
zero_rho	logical; fix actor-partner correlation at zero. Default FALSE
zero_actor	logical; fix actor variance at zero. Default FALSE
zero_partner	logical; fix partner variance at zero. Default FALSE
zero_group	logical; fix group variance at zero. Default FALSE
zero_location	logical; fix location variance at zero. Default FALSE
weight_partners	logical; weight partner dummies by $1/(group_size - 1)$. Default TRUE
fixed_effects	character vector or NULL; fixed effect predictor column names. Default NULL
start	named numeric vector or NULL; optional starting values
maxit	integer; maximum optimizer iterations. Default 200000
stage1_maxit	integer; stage 1 Nelder-Mead iterations. Default 1000
stage2_maxit	integer; stage 2 Nelder-Mead iterations. Default 1000
tol	numeric; convergence tolerance. Default 1e-10
se_method	character; "hessian" (default) or "none"
rho_boundary	numeric; boundary warning threshold. Default 0.99
optimizer	character; "Nelder-Mead" (default) or "L-BFGS-B"
verbose	logical; print progress. Default FALSE
verbose_every	integer; print every N evaluations. Default 50

Value

an object of class "cpsrm"

Reporting partner variance: RAW vs. COMBINED

Because each row sums multiple independent partner effects (its group_size - 1), σ^2_P can be reported on two different, equally valid bases when weight_partners = FALSE:

- **RAW** (σ^2_P itself): the variance of the partner-effect parameter, on the same per-person basis as σ^2_A. Use RAW whenever comparing the magnitude of partner effects to actor effects (e.g. tests of generalized reciprocity).

- **COMBINED** (partner's row multiplied by `group_size - 1`): partner identity's total contribution to the variance of a single observed score. Use COMBINED only when reading a full variance decomposition in which all components must sum to the outcome's total variance.

Mixing the two conventions in the same comparison is a common source of error: because COMBINED mechanically multiplies partner's numerator while RAW does not, a naive COMBINED-vs-RAW comparison can make partner effects look several times more (or less) important than actor effects even when the two are, per person, identical in magnitude. `print.cpsrm`/`summary.cpsrm` report both whenever `weight_partners = FALSE` and group size is constant across the data; when group size varies, or `weight_partners = TRUE` (partner dummies already rescaled to average rather than sum partner effects), only RAW is shown.

Testing variance components: use the boundary-corrected LRT

`cpsrm_run` reports a Wald z-statistic (estimate / SE) for each variance component, but variance components are bounded below by zero, so the usual two-sided reference distribution is not correct for testing whether a variance component is zero. The standard correction (Self & Liang, 1987; Snijders & Bosker, 2012; Stram & Lee, 1994) is to fit the model with and without the component of interest (e.g. `zero_partner = TRUE` vs. `FALSE`), form the likelihood-ratio statistic from each fit's `fit["reml_loglik"]` ($LRT = -2(\ell_{restricted} - \ell_{full})$), and compute a ONE-SIDED p-value:

```
p <- 0.5 * pchisq(LRT, df = 1, lower.tail = FALSE)
```

Testing TWO variance components jointly at their boundary (e.g. actor and partner both zero) is NOT a simple halving of the df = 2 chi-square, because both parameters are boundary-constrained; it requires the three-part mixture (Self & Liang, 1987):

```
p <- 0.25 * pchisq(LRT, df = 0, lower.tail = FALSE) +
     0.50 * pchisq(LRT, df = 1, lower.tail = FALSE) +
     0.25 * pchisq(LRT, df = 2, lower.tail = FALSE)
```

(the df = 0 term is always 0 for any `LRT > 0`). Applying the plain, uncorrected `pchisq` p-value, or halving the df = 2 joint test the same way as a single-component test, both misstate significance and are not valid inferential procedures for this model.

See Also

`create_cp_dummies`

Examples

```
# Prepare dummies using create_cp_dummies(), bind onto data, then call
# cpsrm_run(). See create_cp_dummies() documentation for a full example.
```

cpsrmRun *Deprecated: use* cpsrm_run *instead*

Description

cpsrmRun has been renamed to cpsrm_run. Key argument changes: course.id -> location_id; actor.dummies -> actor_dummies; partner.dummies -> partner_dummies; zero.rho -> zero_rho; zero.actor -> zero_actor; zero.partner -> zero_partner; zero.group -> zero_group; zero.course -> zero_location; weight.partners -> weight_partners; fixed.effects -> fixed_effects; stage1.maxit -> stage1_maxit; stage2.maxit -> stage2_maxit; se.method -> se_method; rho.boundary -> rho_boundary; verbose.every -> verbose_every.

Usage

```
cpsrmRun(
  dv,
  actor.id,
  group.id,
  actor.dummies,
  partner.dummies,
  data,
  course.id = NULL,
  zero.rho = FALSE,
  zero.actor = FALSE,
  zero.partner = FALSE,
  zero.group = FALSE,
  zero.course = FALSE,
  weight.partners = TRUE,
  fixed.effects = NULL,
  stage1.maxit = 1000L,
  stage2.maxit = 1000L,
  se.method = c("hessian", "none"),
  rho.boundary = 0.99,
  verbose.every = 50L,
  ...
)
```

Arguments

dv	string; name of the outcome variable
actor.id	string; name of the actor identifier variable
group.id	string; name of the group identifier variable
actor.dummies	character vector of actor dummy column names
partner.dummies	character vector of partner dummy column names
data	a data.frame

course.id	string or NULL; higher-level clustering variable
zero.rho	logical; fix actor-partner correlation at zero
zero.actor	logical; fix actor variance at zero
zero.partner	logical; fix partner variance at zero
zero.group	logical; fix group variance at zero
zero.course	logical; fix course/location variance at zero
weight.partners	logical; weight partner dummies
fixed.effects	character vector or NULL
stage1.maxit	integer; stage 1 iterations
stage2.maxit	integer; stage 2 iterations
se.method	character; "hessian" or "none"
rho.boundary	numeric; boundary threshold
verbose.every	integer; verbose print frequency
...	additional arguments passed to cpsrm_run

Value

see cpsrm_run

---

create_cp_dummies *Create Actor and Partner Dummy Matrices for the Co-Partner SRM*

Description

Generates the actor and partner dummy matrices required by cpsrm_run. Unlike create_dummies, which produces dummies for the standard Social Relations Model, this function is designed for the Co-Partner SRM where each observation involves one actor and all other members of the group acting simultaneously as partners. Variable group sizes (2, 3, 4, or mixed) are fully supported.

Usage

```
create_cp_dummies(
  data,
  actor_id,
  group_id,
  weight_partners = TRUE,
  prefix_actor = "A",
  prefix_partner = "P"
)
```

Arguments

`data`	a `data.frame` in long format, with one row per person-group observation
`actor_id`	character; column name of the actor/person identifier
`group_id`	character; column name of the group identifier
`weight_partners`	logical; if `TRUE` (default), partner dummies are weighted by $1/(group_size-1)$, making σ^2_P the variance of the average partner contribution. If `FALSE`, dummies are 0/1 and σ^2_P is the variance of the sum of partner contributions. Weighting is recommended when group sizes vary, as it ensures comparability of σ^2_P across group sizes
`prefix_actor`	character; prefix for actor dummy columns. Default `"A"`
`prefix_partner`	character; prefix for partner dummy columns. Default `"P"`

Value

a list with elements:

`actor_mat` matrix of actor dummies (n x n_persons)

`partner_mat` matrix of partner dummies, optionally weighted (n x n_persons)

`actor_names` character vector of actor dummy column names

`partner_names` character vector of partner dummy column names

`group_sizes` integer vector of length n giving the group size for each observation; pass this to `cpsrm_run()` via the `group_sizes` argument

`person_levels` character vector of person IDs (column order)

`n_persons` integer; number of unique persons

`n_groups` integer; number of unique groups

`group_size_table` table of group size frequencies

See Also

`cpsrm_run`, `create_dummies`

Examples

```
# create_cp_dummies() expects one row per PERSON per GROUP, with every
# other group member treated as a simultaneous co-partner (e.g. one row
# per player per team, as in three-person golf teams). sampleDyadData is
# a round-robin/directed-dyad dataset (multiple rows per actor per
# group, one per rated partner) and is NOT the right shape for this
# function -- see create_dummies()/srm_run() for that design instead.
# This example simulates a small, correctly-shaped dataset: 30
# three-person teams drawn from a pool of 90 players.
set.seed(1)
team_dat <- do.call(rbind, lapply(1:30, function(g) {
  data.frame(player = sample(1:90, 3), team = g)
}))
```

```
dummies <- create_cp_dummies(
  data        = team_dat,
  actor_id    = "player",
  group_id    = "team"
)
cat("Persons:", dummies$n_persons, "\n")
cat("Groups:", dummies$n_groups, "\n")
```

`create_dummies` *Create Dummy Variables for the Social Relations Model*

Description

Generates the actor and partner dummy variables required to fit the Social Relations Model (SRM) using multilevel modeling, following the approach of Snijders and Kenny (1999). The function produces N actor dummies and N partner dummies, where N is the maximum group size in the dataset.

Usage

```
create_dummies(
  group_id,
  act_id,
  part_id,
  data,
  include_self = FALSE,
  merge_original = FALSE
)
```

Arguments

`group_id`	string; name of the group identifier variable
`act_id`	string; name of the actor identifier variable
`part_id`	string; name of the partner identifier variable
`data`	a `data.frame` structured in directed dyadic long-form, with one row per ordered (actor, partner) pair
`include_self`	logical; if `TRUE` self-ratings (actor == partner) are retained. Default is `FALSE`
`merge_original`	logical; if `TRUE` the generated identifiers and dummy variables are merged back onto the original dataset and returned together. Default is `FALSE`

Value

a data.frame containing:

pdSRM_act_id integer internal actor identifier

pdSRM_part_id integer internal partner identifier

pdSRM_act_num integer actor slot number within group

pdSRM_part_num integer partner slot number within group

pdSRM_dyad_id integer undirected dyad identifier

a1, a2, ... actor dummy variables

p1, p2, ... partner dummy variables

If merge_original = TRUE, all original variables are appended.

References

Snijders, T. A. B., & Kenny, D. A. (1999). The social relations model for family data: A multilevel approach. *Personal Relationships*, *6*, 471–486. doi:10.1111/j.14756811.1999.tb00204.x

Examples

```
d_out <- create_dummies(
  group_id = "groupId",
  act_id   = "actId",
  part_id  = "partId",
  data     = sampleDyadData
)
head(d_out)
```

---

createDummies *Deprecated: use* create_dummies *instead*

Description

createDummies has been renamed to create_dummies. The argument names have also changed: group.id -> group_id, act.id -> act_id, part.id -> part_id, d -> data, include.self -> include_self, merge.original -> merge_original.

Usage

```
createDummies(
  group.id,
  act.id,
  part.id,
  d,
  include.self = FALSE,
  merge.original = FALSE
)
```

Arguments

group.id	string; name of the group identifier variable
act.id	string; name of the actor identifier variable
part.id	string; name of the partner identifier variable
d	a data.frame in directed dyadic long-form
include.self	logical; retain self-ratings? Default FALSE
merge.original	logical; merge back onto original data? Default FALSE

Value

see create_dummies

---

pdConstruct.pdSRM *Construct pdSRM Object*

Description

Internal method called by pdConstruct to initialize a pdSRM object. Enforces the SRM constraint of equal actor variances, equal partner variances, and a single actor-partner covariance.

Usage

```
## S3 method for class 'pdSRM'
pdConstruct(
  object,
  value = numeric(0),
  form = stats::formula(object),
  nam = nlme::Names(object),
  data = sys.frame(sys.parent()),
  ...
)
```

Arguments

object	an object inheriting from pdSRM
value	an optional initialization value
form	an optional one-sided linear formula
nam	an optional vector of character strings
data	an optional data frame in which to evaluate the variables
...	optional arguments passed to other methods

Value

a pdSRM object representing the SRM positive-definite matrix

Examples

```
d <- createDummies(
  group.id = "groupId", act.id = "actId", part.id = "partId",
  d = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge.original = TRUE
)
o <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
```

---

pdMatrix.pdSRM *Extract Matrix or Square-Root Factor from a pdSRM Object*

Description

Internal method called by pdMatrix to reconstruct the full covariance matrix from the three stored SRM parameters (actor SD, partner SD, actor-partner correlation).

Usage

```
## S3 method for class 'pdSRM'
pdMatrix(object, factor = FALSE)
```

Arguments

object	an object inheriting from pdSRM
factor	logical; if TRUE the upper Cholesky factor is returned, otherwise the full positive-definite matrix

Value

if factor is FALSE, the positive-definite matrix represented by object; if TRUE, an upper triangular Cholesky factor with a logDet attribute

Examples

```
d <- createDummies(
  group.id = "groupId", act.id = "actId", part.id = "partId",
  d = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge.original = TRUE
```

```
)
o <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
```

---

pdSRM *Construct the pdSRM object*

Description

Creates the positive-definite matrix class used to specify the actor-partner covariance structure of the Social Relations Model within lme. This class enforces the SRM constraint that all actors share a single variance, all partners share a single variance, and a single actor-partner covariance (generalized reciprocity) is estimated.

Usage

```
pdSRM(
  value = numeric(0),
  form = NULL,
  nam = NULL,
  data = sys.frame(sys.parent())
)
```

Arguments

value	an optional initialization value, inherited from pdMat
form	an optional one-sided linear formula specifying the row/column names for the matrix
nam	an optional vector of character strings specifying the row/column names for the matrix
data	inherited from the surrounding nlme call

Value

a pdMat object representing a positive-definite matrix conforming to the SRM covariance structure

References

Knight, A. P., & Humphrey, S. E. (2019). Dyadic data analysis. In S. E. Humphrey & J. M. LeBreton (Eds.), *The Handbook for Multilevel Theory, Measurement, and Analysis* (pp. 423–447). American Psychological Association. doi:10.1037/0000115019

Snijders, T. A. B., & Kenny, D. A. (1999). The social relations model for family data: A multilevel approach. *Personal Relationships*, *6*, 471–486. doi:10.1111/j.14756811.1999.tb00204.x

Examples

```
d <- createDummies(
  group.id = "groupId", act.id = "actId", part.id = "partId",
  d = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge.original = TRUE
)
o <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
```

`print.cpsrm` *Print a cpsrm Object*

Description

Print a cpsrm Object

Usage

```
## S3 method for class 'cpsrm'
print(x, digits = 4, ...)
```

Arguments

`x`	an object of class `"cpsrm"`
`digits`	integer; number of digits to print. Default `4`
`...`	additional arguments (currently unused)

Value

`x`, invisibly

`sampleDyadData` *Sample Round Robin Dataset*

Description

A simulated directed dyadic dataset in long format, suitable for demonstrating the functions in this package. The dataset contains observations from a round-robin design in which each person in a group rated every other person. Ratings were collected at two time points.

Usage

```
sampleDyadData
```

Format

A data frame with 1548 rows and 16 variables:

groupId integer; group identification variable

actId integer; actor (rater) identification variable

partId integer; partner (ratee) identification variable

timeId integer; time point identifier (1 or 2)

contact integer; directed dyad-level measure of contact frequency

liking integer; directed dyad-level measure of liking

actEx numeric; actor-level measure of extraversion

actAg numeric; actor-level measure of agreeableness

actMale integer; actor-level binary gender indicator (1 = male)

actAge integer; actor-level age in years

partEx numeric; partner-level measure of extraversion

partAg numeric; partner-level measure of agreeableness

partMale integer; partner-level binary gender indicator (1 = male)

partAge integer; partner-level age in years

groupCohesion numeric; group-level measure of cohesion

groupEfficacy numeric; group-level measure of efficacy

Source

Simulated data created for package demonstration purposes.

`srm_pseudo_rsq` *Calculate Pseudo R-Squared Values for the Social Relations Model*

Description

Computes pseudo R-squared values for each SRM variance component by comparing a null model (intercept only) with a predictor model (with fixed effects). The pseudo R-squared for each component is `(null - predicted) / null`, reflecting the proportion of each variance component explained by the fixed effects.

Usage

```
srm_pseudo_rsq(null_model, predict_model)
```

Arguments

`null_model`	an `lme` object fitted with `pdSRM` and no fixed-effect predictors (intercept only)
`predict_model`	an `lme` object fitted with `pdSRM` and one or more fixed-effect predictors; must use the same dataset and random effects structure as `null_model`

Value

a `data.frame` with three columns and four rows (Group, Actor, Partner, Dyad):

`null` variance component from the null model

`predict` variance component from the predictor model

`pseudoR2` pseudo R-squared: `(null - predict) / null`

Examples

```
d <- create_dummies(
  group_id = "groupId", act_id = "actId", part_id = "partId",
  data = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge_original = TRUE
)
null_mod <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
pred_mod <- nlme::lme(
  liking ~ actEx + partEx + contact,
  random = list(groupId = nlme::pdBlocked(list(
```

```
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
srm_pseudo_rsq(null_model = null_mod, predict_model = pred_mod)
```

`srm_run` *Run the Social Relations Model Using Multilevel Modeling*

Description

A wrapper function that fits the Social Relations Model (SRM) on a directed dyadic dataset using restricted maximum likelihood via `lme`. The function creates the necessary actor and partner dummy variables, constructs the SRM covariance structure using `pdSRM`, and returns both the raw `lme` output and a formatted variance decomposition table.

Usage

```
srm_run(dv, group_id, act_id, part_id, fe_vars = NULL, data)
```

Arguments

`dv`	string; name of the directed dyadic criterion (outcome) variable
`group_id`	string; name of the group identifier variable
`act_id`	string; name of the actor identifier variable
`part_id`	string; name of the partner identifier variable
`fe_vars`	character vector of fixed-effect predictor variable names, or `NULL` (default) for an intercept-only null model
`data`	a `data.frame` at the directed dyad level

Value

a named list with two elements:

`lme.output` the full `lme` model object

`srm.output` a `data.frame` from `srm_var_pct` giving variances, percentages, and reciprocity correlations

References

Knight, A. P., & Humphrey, S. E. (2019). Dyadic data analysis. In S. E. Humphrey & J. M. LeBreton (Eds.), *The Handbook for Multilevel Theory, Measurement, and Analysis* (pp. 423–447). American Psychological Association. doi:10.1037/0000115019

Snijders, T. A. B., & Kenny, D. A. (1999). The social relations model for family data: A multilevel approach. *Personal Relationships*, *6*, 471–486. doi:10.1111/j.14756811.1999.tb00204.x

Examples

```
o <- srm_run(
  dv       = "liking",
  group_id = "groupId",
  act_id   = "actId",
  part_id  = "partId",
  fe_vars  = c("actEx", "partEx", "contact"),
  data     = sampleDyadData[sampleDyadData$timeId == 1, ]
)
o$srm.output
```

`srm_var_pct` *Extract Variance Decomposition from a Fitted SRM*

Description

Extracts and formats the variance components and reciprocity correlations from an `lme` object fitted with the `pdSRM` covariance structure. Returns group, actor, partner, and dyadic (relationship) variances as both raw values and percentages of total variance, along with generalized reciprocity (actor-partner correlation) and dyadic reciprocity.

Usage

```
srm_var_pct(object)
```

Arguments

`object`	an `lme` model object fitted with `pdSRM`

Value

a `data.frame` with two columns and six rows:

`variances.and.covariances` Group, Actor, Partner, and Dyad variances; Generalized Reciprocity covariance; Dyadic Reciprocity covariance

`percents.and.correlations` variance percentages for the four components; Generalized Reciprocity correlation; Dyadic Reciprocity correlation

References

Kenny, D. A., Kashy, D. A., & Cook, W. L. (2006). *Dyadic Data Analysis*. Guilford Press.

Examples

```
d <- create_dummies(
  group_id = "groupId", act_id = "actId", part_id = "partId",
  data = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge_original = TRUE
)
o <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
srm_var_pct(o)
```

---

srmPseudoRSq *Deprecated: use* srm_pseudo_rsq *instead*

Description

srmPseudoRSq has been renamed to srm_pseudo_rsq. The argument names have also changed: null.model -> null_model, predict.model -> predict_model.

Usage

```
srmPseudoRSq(null.model, predict.model)
```

Arguments

null.model	an lme null model fitted with pdSRM
predict.model	an lme predictor model fitted with pdSRM

Value

see srm_pseudo_rsq

srmRun *Deprecated: use* srm_run *instead*

Description

srmRun has been renamed to srm_run. The argument names have also changed: groupId -> group_id, actId -> act_id, partId -> part_id, feVars -> fe_vars.

Usage

```
srmRun(dv, groupId, actId, partId, feVars = NULL, data)
```

Arguments

dv	string; name of the outcome variable
groupId	string; name of the group identifier variable
actId	string; name of the actor identifier variable
partId	string; name of the partner identifier variable
feVars	character vector of fixed-effect predictor names, or NULL
data	a data.frame at the directed dyad level

Value

see srm_run

srmVarPct *Deprecated: use* srm_var_pct *instead*

Description

srmVarPct has been renamed to srm_var_pct.

Usage

```
srmVarPct(object)
```

Arguments

object	an lme model object fitted with pdSRM

Value

see srm_var_pct

summary.cpsrm *Summarize a cpsrm Object*

Description

Summarize a cpsrm Object

Usage

```
## S3 method for class 'cpsrm'
summary(object, ...)
```

Arguments

object	an object of class "cpsrm"
...	additional arguments passed to print.cpsrm

Value

object, invisibly

summary.pdSRM *Summarize a pdSRM Object*

Description

Internal method that produces a summary.pdMat representation of a pdSRM object, used by lme when printing model output.

Usage

```
## S3 method for class 'pdSRM'
summary(object, structName = "Social Relations Model", ...)
```

Arguments

object	an object inheriting from pdSRM
structName	a character string describing the covariance structure; defaults to "Social Relations Model"
...	optional arguments passed to other methods

Value

an object of class summary.pdMat with additional attributes structName and noCorrelation

Examples

```
d <- createDummies(
  group.id = "groupId", act.id = "actId", part.id = "partId",
  d = sampleDyadData[sampleDyadData$timeId == 1, ],
  merge.original = TRUE
)
o <- nlme::lme(
  liking ~ 1,
  random = list(groupId = nlme::pdBlocked(list(
    nlme::pdIdent(~1),
    pdSRM(~ -1 + a1 + a2 + a3 + a4 + p1 + p2 + p3 + p4)
  ))),
  correlation = nlme::corCompSymm(form = ~1 | groupId / pdSRM_dyad_id),
  data = d,
  na.action = stats::na.omit
)
```

Index

∗ **datasets**
sampleDyadData, 18

coef.pdMat, *2*
coef.pdSRM, 2
corMatrix, *3*
corMatrix.pdSRM, 3
cpsrm, 4
cpsrm_run, *4, 5*, 6, *9–11*
cpsrmRun, 9
create_cp_dummies, *4, 5*, *7, 8*, 10
create_dummies, *10, 11*, 12, *13, 14*
createDummies, 13

lme, *16*, *20, 21*, *24*

pdConstruct, *14*
pdConstruct.pdSRM, 14
pdMat, *16*
pdMatrix, *15*
pdMatrix.pdSRM, 15
pdSRM, 16, *19–23*
print.cpsrm, 17, *24*

sampleDyadData, 18
srm_pseudo_rsq, 19, *22*
srm_run, 20, *23*
srm_var_pct, *20*, 21, *23*
srmPseudoRSq, 22
srmRun, 23
srmVarPct, 23
summary.cpsrm, 24
summary.pdSRM, 24