Package: rties (via r-universe)

September 5, 2024

Title Modeling Interpersonal Dynamics

Version 5.0.0

Maintainer Emily Butler <emily.a.butler@gmail.com>

Description The name of this package grew out of our research on temporal interpersonal emotion systems (TIES), hence 'rties'. It provides tools for using a set of models to investigate temporal processes in bivariate (e.g., dyadic) systems. The general approach is to model, one dyad at a time, the dynamics of a variable that is assessed repeatedly from both partners, extract the parameter estimates for each dyad, and then use those parameter estimates as input to a latent profile analysis to extract groups of dyads with qualitatively distinct dynamics. Finally, the profile memberships can be used to either predict, or be predicted by, another variable of interest. Currently, 2 models are supported: 1) inertia-coordination, and 2) a coupled-oscillator. Extended documentation is provided in vignettes. Theoretical background can be found in Butler (2011) <doi:10.1177/1088868311411164> and Butler & Barnard (2019) <doi:10.1097/PSY.0000000000000703>.

Depends R (>= 2.10)

Imports DataCombine, DescTools, deSolve, dplyr, ggplot2, gridExtra, interactions, lattice, lme4, MASS, mclust, nlme, nnet, plyr, zoo

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Suggests devtools, knitr, rmarkdown, sjmisc, sjPlot

VignetteBuilder knitr

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

RemoteUrl https://github.com/ebmtnprof/rties

RemoteRef HEAD

RemoteSha fae56523593318ded0d7d38c8533f39515711dfe

Contents

`actorPartnerDataCross` *Takes individual cross-sectional data from dyads and turns it into actor-partner format.*

Description

Need to use a person ID that has first person in dyad numbered 1-n and second person in dyad = ID + some number larger than the number of dyads. Need dyad ID numbered same as for person ID for the first person in the dyad. Both members in each dyad need to have the same number of rows (rows of missing data are ok)

Usage

```
actorPartnerDataCross(basedata, dyadId, personId)
```

Arguments

`basedata`	A dataframe with cross-sectional dyadic data.
`dyadId`	The name of variable indicating dyad ID.
`personId`	The name of the variable indicating peron ID.

Value

A dataframe in actor-partner format.

Examples

```
data <- rties_ExampleDataShort
newData1 <- data[data$time==1, ] # make a cross-sectional dataframe
newData2 <- actorPartnerDataCross(basedata=newData1, dyadId="couple", personId="couple")
head(newData2)
```

`actorPartnerDataTime` *Takes individual repeated measures data from dyads and turns it into actor-partner format.*

Description

Need to use a person ID that has first person in dyad numbered 1-n and second person in dyad = ID + some number larger than the number of dyads. Need dyad ID numbered same as for person ID for the first person in the dyad. Both members in each dyad need to have the same number of rows (rows of missing data are ok).

Usage

```
actorPartnerDataTime(basedata, dyadId, personId)
```

Arguments

`basedata`	A dataframe with repeated measures dyadic data
`dyadId`	The name of variable indicating dyad ID.
`personId`	The name of the variable indicating peron ID.

Value

A dataframe in actor-partner format.

Examples

```
data <- rties_ExampleDataShort
newData <- actorPartnerDataTime(basedata=data, dyadId="couple", personId="couple")
head(newData)
```

`autoCorPlots` *Produces auto-correlation plots of the observed state variable for lags of -+ 20 time steps for each dyad.*

Description

Produces auto-correlation plots of the observed state variable for lags of -+ 20 time steps for each dyad.

Usage

```
autoCorPlots(basedata, dyadId, personId, obs_name, time_name)
```

Arguments

basedata	A user provided dataframe.
dyadId	The name of the column in the dataframe that has the dyad-level identifier.
personId	The name of the column in the dataframe that has the person-level identifier.
obs_name	The name of the column in the dataframe that has the time-varying observable (e.g., the variable for which dynamics will be assessed).
time_name	The name of the column in the dataframe that indicates sequential temporal observations.

Value

Prints the plots to the screen.

Examples

```
data <- rties_ExampleDataShort
autoCorPlots(basedata=data, dyadId="couple", personId="person", obs_name="dial", time_name="time")
```

---

cloCoupledOde *Provides the equation for a coupled oscillator model for the differential equation solver (ode) to plot*

Description

Provides the equation for a coupled oscillator model for the differential equation solver (ode) to plot

Usage

```
cloCoupledOde(t, state, parameters)
```

Arguments

t	A parameter used by the ode and passed by functions calling cloCoupleOde
state	Another parameter used by the ode and passed by functions calling cloCoupleOde
parameters	Another parameter used by the ode and passed by functions calling cloCoupleOde

Value

A list with the rates of change for each state variable.

`cloPlotTraj` *Plots the bivariate state variable's clo model-predicted temporal trajectories for each latent profile of clo parameters.*

Description

Plots the bivariate state variable's clo model-predicted temporal trajectories for each latent profile of clo parameters.

Usage

```
cloPlotTraj(
  prepData,
  paramEst,
  n_profiles,
  dist0name = NULL,
  dist1name = NULL,
  plot_obs_name = NULL,
  minMax = NULL,
  time_length = NULL,
  printPlots = T
)
```

Arguments

`prepData`	A dataframe that was produced with the "dataPrep" function.
`paramEst`	A dataframe created by indivClo containing the clo parameter estimates for each dyad.
`n_profiles`	The number of latent profiles.
`dist0name`	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
`dist1name`	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1
`plot_obs_name`	An optional name for the observed state variable to appear on plots (e.g., "Emotional Experience").
`minMax`	An optional vector with desired minimum and maximum quantiles to be used for setting the y-axis range on the plots, e.g., minMax <- c(.1, .9) would set the y-axis limits to the 10th and 90th percentiles of the observed state variables. If not provided, the default is to use the minimum and maximum observed values of the state variables.
`time_length`	An optional value specifying how many time points to plot across. Default is the 75th percentile for the observed time variable.
`printPlots`	If true (the default) plots are displayed on the screen.

Value

The function returns the plots as a list.

Examples

```
# See vignettes for examples.
```

`cloResids` *Produces histograms of the residuals from the oscillator model for each dyad.*

Description

Produces histograms of the residuals from the oscillator model for each dyad.

Usage

```
cloResids(derivData, whichModel, printPlots = T)
```

Arguments

`derivData`	A dataframe that was produced with the "estDerivs" function.
`whichModel`	Whether the model to be estimated is the uncoupled-oscillator ("uncoupled") or the coupled-oscillator ("coupled").
`printPlots`	If true (the default) plots are displayed on the screen.

Value

A list with the histograms of the residuals for each dyad.

Examples

```
# See vignettes for examples.
```

`cloUncoupledOde` *Provides the equation for an un-coupled oscillator model for the differential equation solver (ode) to plot*

Description

Provides the equation for an un-coupled oscillator model for the differential equation solver (ode) to plot

Usage

```
cloUncoupledOde(t, state, parameters)
```

Arguments

`t`	A parameter used by the ode and passed by functions calling cloCoupleOde
`state`	Another parameter used by the ode and passed by functions calling cloCoupleOde
`parameters`	Another parameter used by the ode and passed by functions calling cloCoupleOde #'

Value

A list with the rates of change for each state variable.

`crossCorPlots` *Produces cross-correlation plots of the observed state variable for lags of -+ 20 time steps for each dyad.*

Description

Produces cross-correlation plots of the observed state variable for lags of -+ 20 time steps for each dyad.

Usage

```
crossCorPlots(basedata, personId, dyadId, obs_name, time_name)
```

Arguments

`basedata`	A user provided dataframe.
`personId`	The name of the column in the dataframe that has the person-level identifier.
`dyadId`	The name of the column in the dataframe that has the dyad-level identifier.
`obs_name`	The name of the column in the dataframe that has the time-varying observable (e.g., the variable for which dynamics will be assessed).
`time_name`	The name of the column in the dataframe that indicates sequential temporal observations.

Value

Prints the plots to the screen.

Examples

```
data <- rties_ExampleDataShort
crossCorPlots(basedata=data, dyadId="couple", personId="person", obs_name="dial", time_name="time")
```

`dataPrep` *Reformat a user-provided dataframe in a generic form appropriate for* rties *modeling*

Description

The dataframe must be in a specific format and include several specific variables. See the "overview_data_prep" vignette for complete details on the necessary format and follow it closely if you'd like to avoid error messages. That vignette also includes information on how to structure the data if you have two variables within people (rather than two people within dyads) or have indistinguishable dyads.

Usage

```
dataPrep(
  basedata,
  dyadId,
  personId,
  obs_name,
  dist_name,
  time_name,
  time_lag = NULL,
  lagMax = NULL
)
```

Arguments

`basedata`	A user-provided dataframe that includes all variables needed for an rties analysis.
`dyadId`	The name of the column in the dataframe that has the dyad-level identifier.
`personId`	The name of the column in the dataframe that has the person-level identifier.
`obs_name`	The name of the column in the dataframe that has the time-varying observable (e.g., the variable for which dynamics will be assessed).
`dist_name`	The name of the column in the dataframe that has a variable that distinguishes the partners (e.g., sex, mother/daughter, etc) that is numeric and scored 0/1.
`time_name`	The name of the column in the dataframe that indicates sequential temporal observations.

`time_lag` An optional argument for the number of lags for the lagged observable. If a number is provided, the observed variable is lagged that amount. The other option is to use "absMaxCC". In this case the maximum cross-correlation is found for each dyad and the lag at which that occurs is used to lag their observed variables.

`lagMax` An optional argument for the maximum number of lags to be considered for the lagged observable. If a number is provided, it is used as the range of lags to consider. The default is 10*log10(N/m) where N is the number of observations and m the number of series (e.g., the default for the ccf function).

Value

The function returns a dataframe that has all the variables needed for modeling system dynamics, each renamed to a generic variable name, which are:

- id = person id
- dyad = dyad id
- obs = observed state variable
- dist1 = 0/1 variable where the 1's indicate the 1's in the original distinguishing variable
- time = the variable indicating temporal sequence
- dist0 = 0/1 variable where the 1's indicate the 0's in the original distinguishing variable
- obs_deTrend = the observed state variable with each person's linear trend removed
- p_ = all the same variables, but for a person's partner rather than themselves

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person", obs_name="dial",
dist_name="female", time_name="time", time_lag=2)
head(newData)
```

---

dyadByContext

Creates variables indicating dyad by context membership (dyadContext) and person by context (personContext) membership. These are renamed by the dataPrep function into "dyad" and "person," which can be used as normal for all other rties functions that use the dataframe produced by dataPrep. Other functions that use the dataframe prior to the dataPrep step will need to use the dyadContext and personContext variables instead.

Description

Creates variables indicating dyad by context membership (dyadContext) and person by context (personContext) membership. These are renamed by the dataPrep function into "dyad" and "person," which can be used as normal for all other rties functions that use the dataframe produced by dataPrep. Other functions that use the dataframe prior to the dataPrep step will need to use the dyadContext and personContext variables instead.

Usage

```
dyadByContext(
  basedata,
  dyadId,
  personId,
  context,
  obs_name,
  dist_name,
  time_name,
  idConvention
)
```

Arguments

basedata	A user-provided dataframe.
dyadId	The name of the column in the dataframe that has the couple-level identifier.
personId	The name of the column in the dataframe that has the person-level identifier.
context	The name of the column in the dataframe that has the variable indicating context. This must be an integer variable.
obs_name	The name of the column in the dataframe that has the time-varying observable (e.g., the variable for which dynamics will be assessed).
dist_name	The name of the column in the dataframe that has a variable that distinguishes the partners (e.g., sex, mother/daughter, etc) that is numeric and scored 0/1.
time_name	The name of the column in the dataframe that indicates sequential temporal observations. #' @param idConvention The value that was added to the dist1 personID number to get the dist2 personID number

Value

The function returns the original dataframe with additional variables, called "dyadContext" and "personContext." These have the original dyad or person ID numbers with the context number * .001 appended. For example, dyad 33 in context 4 would get the value for "dyadContext" of 33.004. Similarly, person 502 in context 2 would get the value 502.002.

#' @export

---

dyadic *Produces plots for sysVarIn when sysVar is dyadic.*

Description

Produces plots for sysVarIn when sysVar is dyadic.

Usage

```
dyadic(basedata, sysVar_name)
```

Arguments

basedata	A dataframe created internally by the "sysVarInPlots" function.
sysVar_name	The name of the variable in the dataframe that contains the system variable.

Value

A plot with the profiles on the y-axis and the system variable on the x-axis

---

estDerivs *Estimates first and second derivatives of an oberved state variable*

Description

This function makes use of 2 functions written by Steven Boker, "gllaWMatrix" and "gllaEmbed" which are available on his website, http://people.virginia.edu/~smb3u/. It fits a coupled oscillator model for each dyad at different combinations of the input parameters (tau, embeds) and returns the input values and period of oscillation that maximize the R^2 for each dyad. It also estimates first and second derivatives of the observed state variable for each person at the input values that maximize the R^2 for that dyad and returns a dataframe that contains them.

Usage

```
estDerivs(prepData, taus, embeds, delta, idConvention)
```

Arguments

prepData	A dataframe that was produced with the "dataPrep" function.
taus	A vector containing the values of tau to use. Tau indicates the number of time points to lag in the lagged data matrix (see Boker, S.M., Deboeck, P.R., Edler, C., & Keel, P.K. (2010). Generalized local linear approximation of derivatives from time series. In S.M. Chow & E. Ferrer (Eds.), Statistical Methods for Modeling Human Dynamics: An Interdisciplinary Dialogue (pp. 161-178). New York, NY: Taylor & Francis Group). The first derivative is estimated as the mean of the two adjacent slopes across that number of lags, e.g., if tau = 2 then the estimate of the first derivative at time = t is based on the mean of the slopes left and right of time t across 2 observations each. The second derivative is the difference in the two slopes with respect to time. Tau = 1 is sensitive to noise and increasing its value acts as smoothing.
embeds	A vector containing the values of embeds to use. Embeds indicates the number of columns in the lagged data matrix. The minimum = 3 for 2nd order derivatives and higher values increase smoothing.
delta	A value indicating the inter-observation interval. For example, if delta = 2, then every second observation is used in the estimation process.
idConvention	The value that was added to the dist1 ID number to get the dist2 ID number

Value

The function returns a list including: 1) "data" which is a dataframe containing first and second derivative estimates of an observed state variable, and 2) "fitTable" which shows the maximal R^2 achieved for each dyad for a coupled oscillator model, along with the associated tau, embed and estimated period of oscillation.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time")
taus <-c(2,3)
embeds <- c(3,4)
delta <- 1
derivs <- estDerivs(prepData=newData, taus=taus, embeds=embeds, delta=delta, idConvention=500)
head(derivs$fitTable)
summary(derivs$fitTable[ ,4]) # summary of R-square
summary(derivs$fitTable[ ,5]) # summary of period of oscillation
```

---

histAll *Histograms for all numeric variables in a dataframe.*

Description

Useful for checking distributions to assess normality

Usage

```
histAll(basedata)
```

Arguments

basedata	A user-provided dataframe.

Value

No return value. Prints plots to the console.

Examples

```
data <- rties_ExampleDataShort
vars <- c("reltime","ambiv","love","conflict")
newData <- data[vars ]
histAll(newData)
```

`indiv2profilesCat` *Produces plots for sysVarIn when sysVar is categorical and there are 2 profiles*

Description

Produces plots for sysVarIn when sysVar is categorical and there are 2 profiles

Usage

```
indiv2profilesCat(testModel, sysVar0name, sysVar1name)
```

Arguments

`testModel`	The model object created by sysVarIn for the interaction model (e.g., sysVarInteract)
`sysVar0name`	The name created by sysVarInPlots referring to the system variable for partner-0.
`sysVar1name`	The name created by sysVarInPlots referring to the system variable for partner-1.

Value

A plot produced by the interactions package.

`indiv2profilesCont` *Produces plots for sysVarIn when sysVar is continuous and there are 2 profiles*

Description

Produces plots for sysVarIn when sysVar is continuous and there are 2 profiles

Usage

```
indiv2profilesCont(testModel, sysVar0name, sysVar1name)
```

Arguments

`testModel`	The model object created by sysVarIn for the interaction model (e.g., sysVarInteract)
`sysVar0name`	The name created by sysVarInPlots referring to the system variable for partner-0.
`sysVar1name`	The name created by sysVarInPlots referring to the system variable for partner-1.

Value

A plot produced by the interactions package.

`indiv3profilesCat` *Produces plots for sysVarIn when sysVar is categorical and there are 3 profiles*

Description

Produces plots for sysVarIn when sysVar is categorical and there are 3 profiles

Usage

```
indiv3profilesCat(basedata, testModel, sysVar0name, sysVar1name)
```

Arguments

`basedata`	A dataframe created internally by the "sysVarInPlots" function.
`testModel`	The model object created by sysVarIn for the interaction model (e.g., sysVarInteract)
`sysVar0name`	The name created by sysVarInPlots referring to the system variable for partner-0.
`sysVar1name`	The name created by sysVarInPlots referring to the system variable for partner-1.

Value

A list of 3 plots showing the simple slopes for each of the profiles.

`indiv3profilesCont` *Produces plots for sysVarIn when sysVar is continuous and there are 3 profiles*

Description

Produces plots for sysVarIn when sysVar is continuous and there are 3 profiles

Usage

```
indiv3profilesCont(prob, sysVar0name, sysVar1name)
```

Arguments

`prob`	A dataframe created internally by the "sysVarInPlots" function.
`sysVar0name`	The name created by sysVarInPlots referring to the system variable for partner-0.
`sysVar1name`	The name created by sysVarInPlots referring to the system variable for partner-1.

Value

A list of 3 plots showing the simple slopes for each of the profiles.

indiv4profilesCat — *Produces plots for sysVarIn when sysVar is categorical and there are 4 profiles*

Description

Produces plots for sysVarIn when sysVar is categorical and there are 4 profiles

Usage

```
indiv4profilesCat(basedata, testModel, sysVar0name, sysVar1name)
```

Arguments

basedata	A dataframe created internally by the "sysVarInPlots" function.
testModel	The model object created by sysVarIn for the interaction model (e.g., sysVarInteract)
sysVar0name	The name created by sysVarInPlots referring to the system variable for partner-0.
sysVar1name	The name created by sysVarInPlots referring to the system variable for partner-1.

Value

A list of 4 plots showing the simple slopes for each of the profiles.

indiv4profilesCont — *Produces plots for sysVarIn when sysVar is continuous and there are 4 profiles*

Description

Produces plots for sysVarIn when sysVar is continuous and there are 4 profiles

Usage

```
indiv4profilesCont(prob, sysVar0name, sysVar1name)
```

Arguments

prob	A dataframe created internally by the "sysVarInPlots" function.
sysVar0name	The name created by sysVarInPlots referring to the system variable for partner-0.
sysVar1name	The name created by sysVarInPlots referring to the system variable for partner-1.

Value

A list of 4 plots showing the simple slopes for each of the profiles.

`indivClo` *Estimates either an uncoupled or coupled oscillator model for each dyad.*

Description

Both models predict the second derivatives of the observed state variables (with linear trends removed). For the uncoupled oscillator, the predictors are each person's own observed state variables (again with linear trends removed), as well as each person's own first derivatives of the observed state variables (again with linear trends removed. For the coupled oscillator, the predictors are each person's own and partner's observed state variables (again with linear trends removed), as well as each person's own and partner's first derivatives of the observed state variables (again with linear trends removed).

Usage

```
indivClo(derivData, whichModel)
```

Arguments

`derivData`	A dataframe that was produced with the "estDerivs" function.
`whichModel`	Whether the model to be estimated is the "uncoupled" or "coupled" oscillator.

Value

The function returns a list including: 1) the adjusted R^2 for the model for each dyad (called "R2"), and 2) the parameter estimates for the model for each dyad (called "params", for use in either predicting, or being predicted by, the system variable).

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time")
taus <-c(2,3)
embeds <- c(3,4)
delta <- 1
derivs <- estDerivs(prepData=newData, taus=taus, embeds=embeds, delta=delta, idConvention=500)
clo <- indivClo(derivData=derivs$data, whichModel="coupled")
summary(clo$R2)
head(clo$params)
```

`indivCloCompare` *Compares model fit for the uncoupled and coupled oscillator for each dyad's state trajectories using an R-square comparison.*

Description

Fits an uncoupled and coupled oscillator model to each dyad's observed state variables and returns the adjusted R-squares, along with the difference between them (coupled - uncoupled, so positive values indicate better fit for the more complex model).

Usage

```
indivCloCompare(derivData)
```

Arguments

`derivData`	A dataframe that was produced with the "estDerivs" function. #' @examples data <- rties_ExampleDataShort newData <- dataPrep(basedata=data, dyadId="couple", personId="person", obs_name="dial", dist_name="female", time_name="time") taus <-c(2,3) embeds <- c(3,4) delta <- 1 derivs <- estDerivs(prepData=newData, taus=taus, embeds=embeds, delta=delta, idConvention=500) compare <- indivCloCompare(derivData=derivs$data) summary(compare$R2couple)

Value

The function returns a named list including: 1) the adjusted R^2 for the uncoupled model for each dyad (called "R2uncouple"), 2) the adjusted R^2 for the coupled model for each dyad (called "R2couple"), and 3) the difference between the R-squares for each dyad (coupled - uncoupled, called "R2dif").

`indivCloPlots` *Produces plots of either an uncoupled or coupled oscillator model-predicted trajectories overlaid on raw data for each dyad.*

Description

The observed and CLO-model predicted state variables (with linear trends removed) are plotted for each dyad individually.

Usage

```
indivCloPlots(
  derivData,
  whichModel,
  idConvention,
  dist0name = NULL,
  dist1name = NULL,
  plot_obs_name = NULL,
  minMax = NULL,
  printPlots = T
)
```

Arguments

derivData	A dataframe that was produced with the "estDerivs" function.
whichModel	Whether the model to be estimated is the "uncoupled" or "coupled" oscillator.
idConvention	The number that was added to the dist0 partner to get the ID number for the dist1 partner.
dist0name	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
dist1name	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1.
plot_obs_name	An optional name for the observed state variables being plotted (e.g., "Emotional Experience"). Default is observed.
minMax	An optional vector with desired minimum and maximum quantiles to be used for setting the y-axis range on the plots, e.g., minMax <- c(.1, .9) would set the y-axis limits to the 10th and 90th percentiles of the observed state variables. Default is to use the minimum and maximum observed values of the state variables.
printPlots	If true (the default) plots are displayed on the screen.

Value

A list plots of the predicted values against the observed values for each dyad.

Examples

```
# See vignettes for examples.
```

`indivInertCoord` *Estimates versions of the inertia-coordination model for each dyad.*

Description

The user specifies which of 3 models are to be estimated. Each model predicts the observed state variables (with linear trends removed) from either: 1) Inertia only ("inert")- each person's intercept and each person's own observed state variable lagged at the amount specified during the dataPrep step (again with linear trends removed), 2) Coordination only ("coord")- each person's intercept and each person's partner's state variable lagged at the amount specified (again with linear trends removed), or 3) Full inertia-coordination model ("inertCoord") - each person's intercept, each person's own observed state variable lagged at the amount specified during the dataPrep step (again with linear trends removed), and each person's partner's state variable lagged at the amount specified (again with linear trends removed).

Usage

```
indivInertCoord(prepData, whichModel)
```

Arguments

`prepData`	A dataframe that was produced with the "dataPrep" function.
`whichModel`	Whether the model to be estimated is the inertia only model ("inert"), the coordination only model ("coord"), or the full inertia-coordination model ("inertCoord").

Value

The function returns a dataframe containing the parameter estimates, called "params", for use in the latent profile analysis.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
ic <- indivInertCoord(prepData=newData, whichModel="inertCoord")
head(ic$params)
```

indivInertCoordCompare

Compares model fit for the inertia-only, coordination-only and full inertia-coordination model for each dyad's state trajectories using an R-square comparison.

Description

Fits inertia-only, coordination-only and full inertia-coordination models to each dyad's observed state variables and returns the adjusted R-squares, along with the differences between them, so positive values indicate better fit for the first model in the comparison. The 3 comparisons are inertia minus coordination, full model minus inertia, and full model minus coordination.

Usage

```
indivInertCoordCompare(prepData)
```

Arguments

prepData	A dataframe that was produced with the "dataPrep" function.

Value

The function returns a named list including: 1) the adjusted R^2 for the inertia model for each dyad (called "R2inert"), 2) the adjusted R^2 for the coordination model for each dyad (called "R2coord"), 3) the adjusted R^2 for the full inertia-coordination model for each dyad (called "R2inertCoord"), 4) the difference between the R-squares for each dyad for inertia minus coordination (called "R2dif_I_C"), 5) the difference for the full model minus inertia (called "R2dif_IC_I"), and 6) the difference for the full model minus coordination (called "R2dif_IC_C")

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
compare <- indivInertCoordCompare(prepData=newData)
summary(compare$R2inert)
summary(compare$R2coord)
summary(compare$R2inertCoord)
summary(compare$R2dif_IC_I)
```

`indivInertCoordPlots` *Produces plots of the inertia-coordination model-predicted trajectories overlaid on raw data for each dyad.*

Description

The observed state variables (with linear trends removed) are predicted from one of the 3 versions of the inertia-coordination model (inertia only, "inert"; coordination only, "coord"; full inertia-coordination, "inertCoord") for each dyad individually. The predicted trajectories are plotted overlaid on the observed trajectories.

Usage

```
indivInertCoordPlots(
  prepData,
  whichModel,
  dist0name = NULL,
  dist1name = NULL,
  plot_obs_name = NULL,
  minMax = NULL,
  printPlots = T
)
```

Arguments

`prepData`	A dataframe that was produced with the "dataPrep" function.
`whichModel`	Whether the model to be estimated is the inertia only model ("inert"), the coordination only model ("coord"), or the full inertia-coordination model ("inertCoord").
`dist0name`	An optional name for the level-0 of the distinguishing variable to appear on plots (e.g., "Women").
`dist1name`	An optional name for the level-1 of the distinguishing variable to appear on plots (e.g., "Men").
`plot_obs_name`	An optional name for the observed state variable to appear on plots (e.g., "Emotional Experience").
`minMax`	An optional vector with desired minimum and maximum quantiles to be used for setting the y-axis range on the plots, e.g., minMax <- c(.1, .9) would set the y-axis limits to the 10th and 90th percentiles of the observed state variables. If not provided, the default is to use the minimum and maximum observed values of the state variables.
`printPlots`	If true (the default) plots are displayed on the screen.

Value

A list with the plots of the predicted values against the observed values for each dyad.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
temp <- newData[newData$dyad < 5, ]
plots <- indivInertCoordPlots(prepData=temp, whichModel="inertCoord")
```

`inertCoordPlotTraj` *Plots the bivariate state variables' model-predicted temporal trajectories for each latent profile of inertia-coordination parameters.*

Description

Produces sets of prototypical example plots of the state variables' predicted temporal trajectories for each latent profile obtained based on the inertia-coordination parameters. The plots are produced by using the inertia-coordination parameters to predict temporal trajectories, with random noise added at each temporal step.

Usage

```
inertCoordPlotTraj(
  prepData,
  paramEst,
  n_profiles,
  dist0name = NULL,
  dist1name = NULL,
  plot_obs_name = NULL,
  minMax = NULL,
  time_length = NULL,
  numPlots = NULL,
  seed = NULL,
  printPlots = T
)
```

Arguments

`prepData`	A dataframe that was produced with the "dataPrep" function.
`paramEst`	A dataframe created by indivInertCoord containing the inertia-coordination parameter estimates for each dyad.
`n_profiles`	The number of latent profiles.
`dist0name`	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
`dist1name`	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1

plot_obs_name	An optional name for the observed state variable to appear on plots (e.g., "Emotional Experience").
minMax	An optional vector with desired minimum and maximum quantiles to be used for setting the y-axis range on the plots, e.g., minMax <- c(.1, .9) would set the y-axis limits to the 10th and 90th percentiles of the observed state variables. If not provided, the default is to use the minimum and maximum observed values of the state variables.
time_length	An optional value specifying how many time points to plot across. Default is the 75th percentile for the time variable.
numPlots	An optional value controlling how many random examples of each profile are produced. Default is 3.
seed	An optional integer argument that sets the seed of R's random number generator to create reproducible trajectories. If used, the "numPlots" can be set to one - otherwise each plot is replicated 3 times.
printPlots	If true (the default) plots are displayed on the screen.

Value

A list with the plots of predicted trajectories for each dyad.

Examples

```
# See vignettes for examples.
```

---

inertCoordResids *Produces histograms of the residuals from the inertia-coordination model for each dyad.*

Description

Produces histograms of the residuals from the inertia-coordination model for each dyad.

Usage

```
inertCoordResids(prepData, whichModel, printPlots = T)
```

Arguments

prepData	A dataframe that was produced with the "dataPrep" function.
whichModel	Whether the model to be estimated is the inertia only model ("inert"), the coordination only model ("coord"), or the full inertia-coordination model ("inertCoord").
printPlots	If true (the default) plots are displayed on the screen.

Value

A list with the histograms of the residuals for each dyad.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
temp <- newData[newData$dyad < 5, ]
residPlots <- inertCoordResids(prepData=temp, whichModel="inertCoord")
```

`inspectProfiles` *Provides information to help decide how many profiles to use for subsequent rties analyses.*

Description

The function prints out the number of dyads in each profile for a specified number of profiles. It also prints out: 1) a figure showing the best clustering solution as indicated by BIC (e.g., the observed data separated into clusters, produced by mclust), 2) a line plot showing the content of the best solution (e.g., the mean parameter estimates for each profile) and 3) prototypical model-predicted trajectories for each profile. For the inertia-coordination model, it produces sets of prototypical examples by using the inertia-coordination parameters to predict temporal trajectories, with random noise added at each temporal step. This process is required because the inertia-coordination model only represents local dynamics and predictions bear no resemblance to observed variables without the addition of noise. An optional argument, "seed" sets the seed for the random number generator, so you can get the same plots each time. If the "seed" argument is used, then only one plot per profile is produced. For the coupled-oscillator, this step is not necessary and one prototypical trajectory is plotted for each profile.

Usage

```
inspectProfiles(
  whichModel,
  prepData,
  paramEst,
  n_profiles,
  dist0name = NULL,
  dist1name = NULL,
  plot_obs_name = NULL,
  minMax = NULL,
  time_length = NULL,
  numPlots = NULL,
  seed = NULL
)
```

Arguments

whichModel	The name of the model that is being investigated (e.g., "inertCoord" or "clo")
prepData	A dataframe that was produced with the "dataPrep" function.
paramEst	A dataframe created by either indivInertCoord or indivClo containing the parameter estimates for each dyad.
n_profiles	The number of latent profiles.
dist0name	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
dist1name	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1
plot_obs_name	An optional name for the observed state variable to appear on plots (e.g., "Emotional Experience").
minMax	An optional vector with desired minimum and maximum quantiles to be used for setting the y-axis range on the plots, e.g., minMax <- c(.1, .9) would set the y-axis limits to the 10th and 90th percentiles of the observed state variables. If not provided, the default is to use the minimum and maximum observed values of the state variables.
time_length	An optional value specifying how many time points to plot across. Default is the 75th percentile for the time variable.
numPlots	Only relevant for the inertCoord model. An optional value controlling how many random examples of each profile are produced. Default is 3.
seed	Only relevant for the inertCoord model. An optional integer argument that sets the seed of R's random number generator to create reproducible trajectories. If used, the "numPlots" can be set to one - otherwise each plot is replicated 3 times.

Value

A dataframe called "profileData" that contains the profile classification for each dyad.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
 obs_name="dial", dist_name="female", time_name="time")
taus <-c(2,3)
embeds <- c(3,4)
delta <- 1
derivs <- estDerivs(prepData=newData, taus=taus, embeds=embeds, delta=delta, idConvention=500)
clo <- indivClo(derivData=derivs$data, whichModel="coupled")
profiles <- inspectProfiles(whichModel="clo", prepData=newData, paramEst=clo$params, n_profiles=2)
head(profiles)
```

makeBiVarData *Takes typical long-format time-nested-in-person data, and stacks two user-chosen observed variables on top of each other so they can be treated as "bivariate" within person. In other words, two time-series variables from each person are stacked on top of each other, forming a bivariate pair of variables within person (e.g., time in variable in person).*

Description

The resulting data can either: 1) be used with many of the other rties functions (a number of the preparatory functions and plotting will work in this way), but instead of "dyad" being the highest nesting variable, "person" is and should be substituted instead of dyad wherever you would otherwise use dyad, or 2) the data can be reformated again with the "dataPrep" function in rties (see overview_data_prep vignette for information on how to do this), with the resulting data in a format that can be used with any of the other rties functions.

Usage

```
makeBiVarData(
  basedata,
  personId,
  time_name,
  obs1_name,
  obs2_name,
  labels,
  idConvention
)
```

Arguments

basedata	The original dataframe provided by the user that includes at least two time-series variables nested within-person
personId	The name of the column in the dataframe that has the person-level identifier.
time_name	The name of the column in the dataframe that indicates sequential temporal observations.
obs1_name	The name of the column in the dataframe that has the first time-series variable to be stacked. #' @param obs2_name The name of the column in the dataframe that has the second time-series variable to be stacked.
labels	A string vector with the names of the variables that are being stacked.
idConvention	A value that will be added to the varId of the first variable to get the varId for the second variable. It should be a larger value than the highest personId. This varId will then be used by rties in a way similar to personId when partners are nested in dyads.

Value

A dataframe that contains the original data, plus the following columns: 1) var: the names of the stacked variables (taken from "labels"). 2) obs: the stacked observed variable scores, 3) dist: a zero/one distinguishing variable, and 4) varId: a variable ID that is similar to personId for use with rties. The varId for the first stacked variable is the same as the personId, with the varId for the second stacked variable being personId + idConvention.

`makeCrossCorBiVar` *Takes typical time-series wide-format data (e.g., multiple time-varying variables for each person in wide format) and calculates cross-correlations for two user-specified variables within a specified maximum number of lags. It returns a dataframe with the largest absolute cross-correlation and its lag added for each person (e.g., it returns either the most negative or most positive cross-correlation, whichever is larger in absolute terms – the sign is retained).*

Description

Takes typical time-series wide-format data (e.g., multiple time-varying variables for each person in wide format) and calculates cross-correlations for two user-specified variables within a specified maximum number of lags. It returns a dataframe with the largest absolute cross-correlation and its lag added for each person (e.g., it returns either the most negative or most positive cross-correlation, whichever is larger in absolute terms – the sign is retained).

Usage

```
makeCrossCorBiVar(basedata, personId, obs1_name, obs2_name, maxLag)
```

Arguments

`basedata`	The original dataframe provided by the user that includes at least two time-series variables nested within-person
`personId`	The name of the column in the dataframe that has the person-level identifier.
`obs1_name`	The name of the column in the dataframe that has the first time-series variable to be stacked.
`obs2_name`	The name of the column in the dataframe that has the second time-series variable to be stacked.
`time_name`	The name of the column in the dataframe that indicates sequential temporal observations.

Value

A cross-sectional version of the original dataframe with maximal absolute-value cross-correlations and their lags added for each person.

makeCrossCorDyadic	*Calculates cross-correlations for a variable that is nested by person within dyad (e.g. it is the same variable for both partners). It returns a dataframe with either: 1) if "time_lag" is null, the largest absolute cross-correlation and its lag added for each dyad (e.g., it returns either the most negative or most positive cross-correlation, whichever is larger in absolute terms – the sign is retained), or 2) if "time_lag is specified, the cross-correlations for each dyad at that lag.*

Description

Calculates cross-correlations for a variable that is nested by person within dyad (e.g. it is the same variable for both partners). It returns a dataframe with either: 1) if "time_lag" is null, the largest absolute cross-correlation and its lag added for each dyad (e.g., it returns either the most negative or most positive cross-correlation, whichever is larger in absolute terms – the sign is retained), or 2) if "time_lag is specified, the cross-correlations for each dyad at that lag.

Usage

```
makeCrossCorDyadic(
  basedata,
  dyadId,
  personId,
  obs_name,
  dist_name,
  time_lag = NULL,
  lagMax = NULL
)
```

Arguments

basedata	The original dataframe provided by the user that includes all variables needed for an rties analysis, including potential system and control variables, etc.
dyadId	The name of the column in the dataframe that has the couple-level identifier.
personId	The name of the column in the dataframe that has the person-level identifier.
obs_name	The name of the column in the dataframe that has the time-varying observable (e.g., the variable for which dynamics will be assessed).
dist_name	The name of the column in the dataframe that has a variable that distinguishes the partners (e.g., sex, mother/daughter, etc) that is numeric and scored 0/1.
time_lag	If null (the default), the maximum absolute value cross-correlation and its corresponding lag are returned. Otherwise, the cross-correlation at the specified time lag is returned.
lagMax	Maximum lag at which to calculate the acf if time_lag is null. Default is 10*log10(N/m) where N is the number of observations and m the number of series.

Value

A cross-sectional version of the original dataframe with maximal absolute-value cross-correlations and their lags added for each dyad.

Examples

```
data <- rties_ExampleDataShort
newData <- makeCrossCorr(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female")
head(newData)
```

`makeDist` *Create a distinguishing variable (called "dist") for non-distinguishable dyads by assigning the partner who is lower on a chosen variable a 0 and the partner who is higher on the variable a 1.*

Description

Create a distinguishing variable (called "dist") for non-distinguishable dyads by assigning the partner who is lower on a chosen variable a 0 and the partner who is higher on the variable a 1.

Usage

```
makeDist(basedata, dyadId, personId, time_name, dist_name)
```

Arguments

`basedata`	A user-provided dataframe.
`dyadId`	The name of the column in the dataframe that has the couple-level identifier.
`personId`	The name of the column in the dataframe that has the person-level identifier.
`time_name`	The name of the column in the dataframe that indicates sequential temporal observations.
`dist_name`	The name of the column in the dataframe that holds the variable to use for distinguishing the partners. For example, if "influence" was the variable, for each dyad the partner scoring lower on "influence" would be given a score of 0 on "dist" and the partner scoring higher on "influence" would be given a score of 1 on "dist"

Value

The function returns the original dataframe with an additional variable, called "dist" that distinguishes between partners based on the user-specified variable indicated by "dist_name"

Examples

```
data <- rties_ExampleDataShort
newData <- makeDist(basedata=data, dyadId="couple", personId="person",
time_name="time", dist_name="relstress")
summary(newData$dist)
```

`makeFullData` *Combines profile membership data from the latent profile analysis with other data for using the profile membership to predict and be predicted by the system variable.*

Description

Combines profile membership data from the latent profile analysis with other data for using the profile membership to predict and be predicted by the system variable.

Usage

```
makeFullData(basedata, dyadId, personId, dist_name, lpaData)
```

Arguments

`basedata`	The original dataframe provided by the user that includes all variables needed for an rties analysis, including potential system and control variables, etc.
`dyadId`	The name of the column in the dataframe that has the couple-level identifier.
`personId`	The name of the column in the dataframe that has the person-level identifier.
`dist_name`	The name of the column in the dataframe that has a variable that distinguishes the partners (e.g., sex, mother/daughter, etc) that is numeric and scored 0/1.
`lpaData`	The object returned by the "inspectProfiles" function

Value

A dataframe that contains all variables needed for using the profiles to predict, or be predicted by, the system variable.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
ic <- indivInertCoord(prepData=newData, whichModel="inertCoord")
profiles <- inspectProfiles(whichModel="inertCoord", prepData=newData,
paramEst=ic$params, n_profiles=2)
fullData <- makeFullData(basedata=data, dyadId="couple", personId="person",
dist_name="female", lpaData=profiles)
head(fullData)
```

Max_Min_CCF_Signed *A helper function for makeCrossCorr*

Description

A helper function for makeCrossCorr

Usage

```
Max_Min_CCF_Signed(a, b, lagMax)
```

Arguments

a	First time-series used in the cross-correlation
b	Second time-series used in the cross-correlation
lagMax	Maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series.

Value

A list of maximum absolute value cross-correlations and the lag at which they occurred.

plotDataByProfile *Plots of de-trended observed variable over time, with dyads separated into groups based on LPA profile membership.*

Description

Plots of de-trended observed variable over time, with dyads separated into groups based on LPA profile membership.

Usage

```
plotDataByProfile(
  prepData,
  fullData,
  n_profiles,
  dist0name = NULL,
  dist1name = NULL,
  plot_obs_name = NULL,
  printPlots = T
)
```

Arguments

prepData	A dataframe created by the dataPrep function.
fullData	A dataframe created by the makeFullData function.
n_profiles	The number of profiles that were estimated.
dist0name	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
dist1name	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1.
plot_obs_name	An optional name for the observed state variable to appear on plots (e.g., "Emotional Experience").
printPlots	If true (the default) plots are displayed on the screen.

Value

A list of plots.

Examples

```
# See vignettes for examples.
```

---

plotRaw *Plots of observed variable over time by dyad.*

Description

Produces plots of the observed variable for each dyad over time to check for data errors, etc.

Usage

```
plotRaw(
  basedata,
  dyadId,
  obs_name,
  dist_name,
  time_name,
  dist0name = NULL,
  dist1name = NULL,
  plot_obs_name = NULL,
  printPlots = T
)
```

Arguments

basedata	A user provided dataframe.
dyadId	The name of the column in the dataframe that has the dyad-level identifier.
obs_name	The name of the column in the dataframe that has the time-varying observable (e.g., the variable for which dynamics will be assessed).
dist_name	The name of the column in the dataframe that has a variable that distinguishes the partners (e.g., sex, mother/daughter, etc) that is numeric and scored 0/1.
time_name	The name of the column in the dataframe that indicates sequential temporal observations.
dist0name	An optional name for the level-0 of the distinguishing variable to appear on plots (e.g., "Women").
dist1name	An optional name for the level-1 of the distinguishing variable to appear on plots (e.g., "Men").
plot_obs_name	An optional name for the observed state variable to appear on plots (e.g., "Emotional Experience").
printPlots	If true (the default) plots are displayed on the screen.

Value

A list of plots.

Examples

```
data <- rties_ExampleDataShort
plotRaw(basedata=data, dyad="couple", obs_name="dial", dist_name="female", time_name="time")
```

---

removeDyads *Remove data for specified dyads from a dataframe*

Description

Useful for cleaning data if some dyads have extensive missing or otherwise problematic data. The function will automatically remove dyads where the two partners have an unequal number of observations or completely missing data. If this happens it provides a warning, prints the dyads that were removed and returns two lists (unEqual and dyadsMissing) with the dyad ids. In addition, you can 1) provide a vector of dyad IDs to remove, or 2) provide a time cut to remove dyads with fewer time points than the cutoff.

Usage

```
removeDyads(
  basedata,
  dyadId,
  dist_name,
  obs_name,
  dyadsToCut = NULL,
  timeCut = NULL
)
```

Arguments

basedata	A user provided dataframe.
dyadId	The variable in the dataframe specifying dyad ID.
dist_name	The name of the column in the dataframe that has a variable that distinguishes the partners (e.g., sex, mother/daughter, etc) that is numeric and scored 0/1.
obs_name	The name of the column in the dataframe that has the time-varying observable (e.g., the variable for which dynamics will be assessed).
dyadsToCut	A vector of dyad IDs to remove.
timeCut	The time value cutoff for removing dyads with fewer time points.

Value

A list with 1) a dataframe (basedata) with the data removed for dyads where the partners have unequal numbers of observations or completely missing data, any dyads specified by dyadsToCut, and dyads who had observations less than or equal to the timeCut value, 2) a list of the dyad IDs with unequal observations (unEqual) and 3) a list of the dyad IDs with completely missing observations (dyadsMissing).

Examples

```
data <- rties_ExampleDataShort
dyadsToCut <- c(3, 12)
newData <- removeDyads(basedata=data, dyadId="couple", obs_name="dial", dist_name="female", dyadsToCut=dyadsToCu
```

---

rties_ExampleDataFull *Data for examples in the vignettes.*

Description

A dataset containing variables for the examples in the vignettes.

Usage

```
rties_ExampleDataFull
```

Format

An object of class `data.frame` with 78858 rows and 19 columns.

`rties_ExampleDataShort`

Data for the function examples.

Description

A dataset containing variables for the examples of the functions.

Usage

```
rties_ExampleDataShort
```

Format

An object of class `data.frame` with 17884 rows and 19 columns.

`rties_ExampleData_Demo`

Data for demonstrating rties models.

Description

A dataset containing a minimal set of variables for demonstrating rties analyses.

Usage

```
rties_ExampleData_Demo
```

Format

An object of class `data.frame` with 12 rows and 6 columns.

signAbsMaxCC *A helper function for makeCrossCorBiVar*

Description

A helper function for makeCrossCorBiVar

Usage

```
signAbsMaxCC(a, b, maxLag)
```

Arguments

a	First time-series used in the cross-correlation
b	Second time-series used in the cross-correlation
lagMax	Maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series.

Value

A list of maximum absolute value cross-correlations and the lag at which they occurred.

smoothData *Smooth one column of a dataframe that has time nested in people (e.g., the function is applied one person at a time) with a user specified smoothing window*

Description

Smooth one column of a dataframe that has time nested in people (e.g., the function is applied one person at a time) with a user specified smoothing window

Usage

```
smoothData(basedata, window, personId, obs_name)
```

Arguments

basedata	A user provided dataframe.
window	A number specifying the window size to be smoothed over.
personId	The variable in the dataframe specifying person ID.
obs_name	The name of the column in the dataframe that has the time-varying observable (e.g., the variable that will be smoothed).

Value

The original dataframe with an additional column holding the smoothed observation variable, called "obsSmooth"

`sysVarIn` *Provides results for predicting couples' latent profile membership from the system variable.*

Description

If there are 2 profiles, then glm binomial regression models are used. If there are more than 2 profiles then multinomial regression is used (from the nnet package). The system variable can be either dyadic (sysVarType = "dyadic"), where both partners have the same score (e.g., relationship length) or individual (sysVarType = "indiv"), where the partners can have different scores (e.g., age). For dyadic system variables, a couple's shared score is the only predictor of their profile membership (called "sysVar"). For individual system variables, two models are tested, one with the main effects of both partner's system variable ("sysVarMain") and one with the main effects and their interaction ("sysVarInteract"). In both cases an intercept-only model is included as a comparison point (called "base"). The function returns a list of the full model results.

Usage

```
sysVarIn(fullData, sysVar_name, sysVarType, n_profiles)
```

Arguments

`fullData`	A dataframe created by the makeFullData function.
`sysVar_name`	The name of the variable in the dataframe that contains the system variable to be predicted by profile membership.
`sysVarType`	Whether the system variable is "dyadic", which means both partners have the same score, or "indiv" which means the partners can have different scores
`n_profiles`	The number of latent profiles.

Value

A list including the glm or multinom objects containing the full results for each model (called "models").

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
ic <- indivInertCoord(prepData=newData, whichModel="inertCoord")
profiles <- inspectProfiles(whichModel="inertCoord", prepData=newData,
paramEst=ic$params, n_profiles=2)
fullData <- makeFullData(basedata=data, dyadId="couple", personId="person",
```

```
dist_name="female", lpaData=profiles, params=ic$params)
sysIn <- sysVarIn(fullData=fullData, sysVar_name="conflict", sysVarType="indiv", n_profiles=2)
summary(sysIn$models$sysVarMain)
```

`sysVarInPlots` *Produces plots for interpreting the results from sysVarIn.*

Description

Produces plots for interpreting the results from sysVarIn.

Usage

```
sysVarInPlots(
  fullData,
  sysVar_name,
  sysVarType,
  n_profiles,
  testModel = NULL,
  dist0name = NULL,
  dist1name = NULL,
  printPlots = T
)
```

Arguments

`fullData`	A dataframe created by the "makeFullData" function.
`sysVar_name`	The name of the variable in the dataframe that contains the system variable.
`sysVarType`	Whether the system variable is "dyadic", which means both partners have the same score, or "indiv" which means the partners can have different scores
`n_profiles`	The number of latent profiles.
`testModel`	The name of the model that is being interpreted (e.g., sysIn$models$sysVarInteract). Only needed when the system variable is "indiv" (e.g., individual scores for each partner)
`dist0name`	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
`dist1name`	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1
`printPlots`	If true (the default) plots are displayed on the screen.

Value

Single plots or a list of plots (depending on the model that is being interpreted).

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
ic <- indivInertCoord(prepData=newData, whichModel="inertCoord")
profiles <- inspectProfiles(whichModel="inertCoord", prepData=newData,
paramEst=ic$params, n_profiles=2)
fullData <- makeFullData(basedata=data, dyadId="couple", personId="person",
dist_name="female", lpaData=profiles, params=ic$params)
sysIn <- sysVarIn(fullData=fullData, sysVar_name="conflict", sysVarType="indiv", n_profiles=2)
sysVarInPlots(fullData=fullData, sysVar_name="conflict", sysVarType="indiv",
n_profiles=2, testModel=sysIn$models$sysVarInteract)
```

---

sysVarInResults *Produces results from sysVarIn.*

Description

Produces results from sysVarIn.

Usage

```
sysVarInResults(baseModel, testModel, n_profiles)
```

Arguments

baseModel	The name of the model that was produced by sysVarIn to be used as the null model for comparison (e.g., sysIn$models$base).
testModel	The name of the model that was produced by sysVarIn that you want results for (e.g., sysIn$models$sysVarMain or sysIn$models$sysVarInteract).
n_profiles	The number of latent profiles. #' @examples data <- rties_ExampleDataShort newData <- dataPrep(basedata=data, dyadId="couple", personId="person", obs_name="dial", dist_name="female", time_name="time", time_lag=2) ic <- indivInertCoord(prepData=newData, whichModel="inertCoord") profiles <- inspectProfiles(whichModel="inertCoord", prepData=newData, paramEst=ic$params, n_profiles=2) fullData <- makeFull-Data(basedata=data, dyadId="couple", personId="person", dist_name="female", lpaData=profiles, params=ic$params) sysIn <- sysVarIn(fullData=fullData, sys-Var_name="conflict", sysVarType="indiv", n_profiles=2) sysVarInResults(baseModel=sysIn$models$bas testModel=sysIn$models$sysVarMain, n_profiles=2)

Value

A list of results including a comparison of the test model to the null (either a LRT or Chisquare test depending on the model), a summary of the parameter estimates, exponentiated parameter estimates (e.g., odds ratios), and p values for the parameter estimates.

`sysVarOut` *Provides results for predicting the system variable from the latent profiles of the dynamic parameters.*

Description

The system variable can be either dyadic (sysVarType = "dyadic"), where both partners have the same score (e.g., relationship length) or individual (sysVarType = "indiv"), where the partners can have different scores (e.g., age). For dyadic system variables, the only predictor is profile membership and the model is a regular regression model since all variables are at the level of the dyad. If the system variable is individual then the model is a random-intercept dyadic model and 3 models are estimated: 1) the main effect of profile membership, 2) main effects of profile membership and the distinguishing variable, and 3) the interaction of profile membership and the distinguishing variable. If the system variable is not normally distributed, any of the generalized linear models supported by glm (for dyadic system variables) or glmer (for individual system variables) are available by specifying the "family" distribution.

Usage

```
sysVarOut(
  fullData,
  sysVar_name,
  sysVarType,
  dist0name = NULL,
  dist1name = NULL,
  family = NULL
)
```

Arguments

`fullData`	A dataframe created by the "makeFullData" function.
`sysVar_name`	The name of the variable in the dataframe that contains the system variable to be predicted by profile membership.
`sysVarType`	Whether the system variable is "dyadic", which means both partners have the same score, or "indiv" which means the partners can have different scores
`dist0name`	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
`dist1name`	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1
`family`	An optional argument specifying the error distribution and link function to be used in the model. Any of the "family" options supported by glm (for dyadic system variables) or glmer (for individual system variables) are available. Default is gaussian.

Value

For normally distributed system variables, the function returns a list including the lm or lme objects containing the full results for each model (called "models"). Similarly, for non-normal system variables, the function returns a list of the glm or glmer objects containing the full results for the models.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
ic <- indivInertCoord(prepData=newData, whichModel="inertCoord")
profiles <- inspectProfiles(whichModel="inertCoord", prepData=newData,
paramEst=ic$params, n_profiles=2)
fullData <- makeFullData(basedata=data, dyadId="couple", personId="person",
dist_name="female", lpaData=profiles, params=ic$params)
sysOut <- sysVarOut(fullData=fullData, sysVar_name="conflict", sysVarType="indiv")
summary(sysOut$models$profilePlusDist)
```

---

sysVarOutPlots *Produces plots for interpreting the results from sysVarIn.*

Description

Produces plots for interpreting the results from sysVarIn.

Usage

```
sysVarOutPlots(
  fullData,
  sysVar_name,
  sysVarType,
  testModel,
  dist0name = NULL,
  dist1name = NULL,
  binomial = F
)
```

Arguments

fullData	A dataframe created by the "makeFullData" function.
sysVar_name	The name of the variable in the dataframe that contains the system variable.
sysVarType	Whether the system variable is "dyadic", which means both partners have the same score, or "indiv" which means the partners can have different scores
testModel	The name of the model that is being interpreted (e.g., sysIn$models$sysVarInteract).

dist0name	An optional name for the level-0 of the distinguishing variable (e.g., "Women"). Default is dist0.
dist1name	An optional name for the level-1 of the distinguishing variable (e.g., "Men"). Default is dist1
binomial	Whether the system variable is binomial. Default is false.

Value

Single plots or a list of plots (depending on the model that is being interpreted).

Examples

```
# See vignettes for examples.
```

`sysVarOutResults` *Produces results from sysVarOut.*

Description

Produces results from sysVarOut.

Usage

```
sysVarOutResults(baseModel, testModel, Gaussian = TRUE)
```

Arguments

baseModel	The name of the model that was produced by sysVarOut to be used as the null model for comparison (e.g., sysOut$models$base).
testModel	The name of the model that was produced by sysVarOut that you want results for (e.g., sysOut$models$profile, sysOut$models$profilePlusDist, sysOut$models$profileByDist).
Gaussian	Whether the system variable is Gaussian. Default is true.

Value

A list of results including an LRT comparison of the test model to the null, an omnibus anova test for the parameters in the model (this is identical to the LRT test for Gaussian dyadic system variables), a summary of the parameter estimates, and exponentiated parameter estimates (e.g., odds ratios) if Gaussian = FALSE.

Examples

```
data <- rties_ExampleDataShort
newData <- dataPrep(basedata=data, dyadId="couple", personId="person",
obs_name="dial", dist_name="female", time_name="time", time_lag=2)
ic <- indivInertCoord(prepData=newData, whichModel="inertCoord")
profiles <- inspectProfiles(whichModel="inertCoord", prepData=newData,
paramEst=ic$params, n_profiles=2)
fullData <- makeFullData(basedata=data, dyadId="couple", personId="person",
dist_name="female", lpaData=profiles, params=ic$params)
sysOut <- sysVarOut(fullData=fullData, sysVar_name="conflict", sysVarType="indiv")
sysVarOutResults(baseModel=sysOut$models$base, testModel=sysOut$models$profileByDist)
```

Index