Package: dySEM (via r-universe)

September 10, 2024

Title Dyadic Structural Equation Modeling

Version 1.1.0

Description Scripting of structural equation models via 'lavaan' for Dyadic Data Analysis, and helper functions for supplemental calculations, tabling, and model visualization. Current models supported include Dyadic Confirmatory Factor Analysis, the Actor–Partner Interdependence Model (observed and latent), the Common Fate Model (observed and latent), Mutual Influence Model (latent), and the Bifactor Dyadic Model (latent).

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/jsakaluk/dySEM, https://jsakaluk.github.io/dySEM/

BugReports https://github.com/jsakaluk/dySEM/issues

Imports dplyr, gt, lavaan, lifecycle, magrittr, rlang, semPlot, stringr, tibble

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

LazyData true

Depends R (>= 2.10)

Config/testthat/edition 3

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

RemoteUrl https://github.com/jsakaluk/dysem

RemoteRef HEAD

RemoteSha b8eab4594ef70f7bece2849544c7397e9bdbd32d

Contents

`commitmentM` *Ratings of relational satisfaction and commitment from 282 (M)ixed-sex couples*

Description

A data set containing 5 ratings of satisfaction and 5 ratings of commitment for each member of a mixed-sex romantic dyad. Measured using the Investment Model Scale subscales (Rusbult, Martz, & Agnew, 1998). Data are from Sakaluk, Fisher, and Kilshaw's (2021) study of dyadic invariance. Variable names have been re-coded to follow a stem-item-partner ("sip") order, with a delimiter ("_") between the item number and partner distinguishing character.

Usage

```
data(commitmentM)
```

Format

A data frame with 282 rows and 20 variables:

sat.g1_f Satisfaction item 1 for female partner

sat.g2_f Satisfaction item 2 for female partner

sat.g3_f Satisfaction item 3 for female partner

sat.g4_f Satisfaction item 4 for female partner

sat.g5_f Satisfaction item 5 for female partner

com1_f Commitment items item 1 for female partner

com2_f Commitment items item 2 for female partner

com3_f Commitment items item 3 for female partner

com4_f Commitment items item 4 for female partner

com5_f Commitment items item 5 for female partner

sat.g1_m Satisfaction item 1 for male partner

sat.g2_m Satisfaction item 2 for male partner

sat.g3_m Satisfaction item 3 for male partner

sat.g4_m Satisfaction item 4 for male partner

sat.g5_m Satisfaction item 5 for male partner

com1_m Commitment items item 1 for male partner

com2_m Commitment items item 2 for male partner

com3_m Commitment items item 3 for male partner

com4_m Commitment items item 4 for male partner

com5_m Commitment items item 5 for male partner

References

Sakaluk, J. K., Fisher, A. N., & Kilshaw, R. E.(2021). Dyadic measurement invariance and its importance for replicability in romantic relationship research. *Personal Relationships*, *28*(1), 190-226. .

`commitmentQ` *Ratings of relational satisfaction and commitment from 282 (Q)ueer couples*

Description

A data set containing 5 ratings of satisfaction and 5 ratings of commitment for each member of a dyad in which one or more members identify as LGBTQ+. Measured using the Investment Model Scale subscales (Rusbult, Martz, & Agnew, 1998). Data are from Sakaluk, Fisher, and Kilshaw (2021). Variable names follow a stem-partner-item ("spi") order, with a delimiter (".") between the stem and distinguishing partner character, and another delimiter ("_") between the distinguishing partner character and item number.

Usage

```
data(commitmentQ)
```

Format

A data frame with 118 rows and 20 variables:

sat.g.1_1 Satisfaction item 1 for partner1

sat.g.1_2 Satisfaction item 2 for partner1

sat.g.1_3 Satisfaction item 3 for partner1

sat.g.1_4 Satisfaction item 4 for partner1

sat.g.1_5 Satisfaction item 5 for partner1

com.1_1 Commitment items item 1 for partner1

com.1_2 Commitment items item 2 for partner1

com.1_3 Commitment items item 3 for partner1

com.1_4 Commitment items item 4 for partner1

com.1_5 Commitment items item 5 for partner1

sat.g.2_1 Satisfaction item 1 for partner 2

sat.g.2_2 Satisfaction item 2 for partner 2

sat.g.2_3 Satisfaction item 3 for partner 2

sat.g.2_4 Satisfaction item 4 for partner 2

sat.g.2_5 Satisfaction item 5 for partner 2

com.2_1 Commitment items item 1 for partner 2

com.2_2 Commitment items item 2 for partner 2

com.2_3 Commitment items item 3 for partner 2

com.2_4 Commitment items item 4 for partner 2

com.2_5 Commitment items item 5 for partner 2

References

Sakaluk, J. K., Fisher, A. N., & Kilshaw, R. E.(2021). Dyadic measurement invariance and its importance for replicability in romantic relationship research. *Personal Relationships*, *28*(1), 190-226. .#'

DRES *Relationship quality and sexual satisfaction of 121 couples*

Description

A dataset containing 9 observed indicators of relationship quality (PRQC) and 5 observed indicators of sexual satisfaction from 121 couples in the DRES (Daily Relationship Experiences Study; Raposo, Impett, & Muise, in press)

Usage

```
data(DRES)
```

Format

A data frame with 121 rows and 28 variables:

PRQC_1.1 PRQC item 1 for partner 1

PRQC_2.1 PRQC item 2 for partner 1

PRQC_3.1 PRQC item 3 for partner 1

PRQC_4.1 PRQC item 4 for partner 1

PRQC_5.1 PRQC item 5 for partner 1

PRQC_6.1 PRQC item 6 for partner 1

PRQC_7.1 PRQC item 7 for partner 1

PRQC_8.1 PRQC item 8 for partner 1

PRQC_9.1 PRQC item 9 for partner 1

PRQC_1.2 PRQC item 1 for partner 2

PRQC_2.2 PRQC item 2 for partner 2

PRQC_3.2 PRQC item 3 for partner 2

PRQC_4.2 PRQC item 4 for partner 2

PRQC_5.2 PRQC item 5 for partner 2

PRQC_6.2 PRQC item 6 for partner 2

PRQC_7.2 PRQC item 7 for partner 2

PRQC_8.2 PRQC item 8 for partner 2

PRQC_9.2 PRQC item 9 for partner 2

sexsat1.1 sexual satisfaction item 1 for partner 1

sexsat2.1 sexual satisfaction item 2 for partner 1

sexsat3.1 sexual satisfaction item 3 for partner 1

sexsat4.1 sexual satisfaction item 4 for partner 1

sexsat5.1 sexual satisfaction item 5 for partner 1

sexsat1.2 sexual satisfaction item 1 for partner 2

sexsat2.2 sexual satisfaction item 2 for partner 2

sexsat3.2 sexual satisfaction item 3 for partner 2

sexsat4.2 sexual satisfaction item 4 for partner 2

sexsat5.2 sexual satisfaction item 5 for partner 2

References

Raposo, S., Impett, E. A., & Muise, A. (2020). Avoidantly Attached Individuals Are More Exchange-Oriented and Less Communal in the Bedroom. *Archives of Sexual Behavior*, *49*, 2863–2881. https://doi.org/10.1007/s10508-020-01813-9

`getConstraintTests` *A function that performs a score test for relaxing each invariance equality constraint between partners in a given dyadic SEM model.*

Description

A function that performs a score test for relaxing each invariance equality constraint between partners in a given dyadic SEM model.

Usage

```
getConstraintTests(constrainFit, filterSig = FALSE)
```

Arguments

`constrainFit`	fitted lavaan model with dyadic invariance equality constraints
`filterSig`	logical indicating whether to filter for significant constraints (default is FALSE)

Value

a data frame with rows of equality constraints (now with readable param labels) and test statistic, df, and p for whether constraint worsens model fit

Examples

```
dvn <- scrapeVarCross(dat = commitmentM, x_order = "sip", x_stem = "sat.g",
x_delim2="_", distinguish_1="f", distinguish_2="m")

sat.resids.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = c("loadings", "intercepts", "residuals"),
constr_dy_struct = "none")

sat.resids.mod <- lavaan::cfa(sat.resids.script, data = commitmentM, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = TRUE)

getConstraintTests(sat.resids.mod)
```

`getDydmacs` *Calculates dmacs difference in expected indicator scores for between dyad members*

Description

Calculates dmacs difference in expected indicator scores for between dyad members

Usage

```
getDydmacs(dat, dvn, fit, nodewidth = 0.01, lowerLV = -5, upperLV = 5)
```

Arguments

dat	data frame of indicators
dvn	input dvn list from scrapeVarCross
fit	outputted dyadic cfa lavaan object; should be from a partial-invariance model
nodewidth	space between nodes during quadrature approximation (default = .01)
lowerLV	lowest latent variable value evaluated (default = -5)
upperLV	greatest latent variable value evaluated (default = 5)

Value

vector of d_macs values

See Also

Other supplemental model calculators: `getDyReliability()`, `getIndistFit()`

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2")
sat.config.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = "none",
constr_dy_struct = "none")

sat.config.mod <- lavaan::cfa(sat.config.script, data = commitmentQ, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = TRUE)
getDydmacs(commitmentQ, dvn, sat.config.mod)
```

---

getDyReliability *A Function Calculates Omega Total Coefficients from a Dyadic CFA*

Description

This function takes the model from fitted scriptCFA() scripts and returns omega total coefficients for each dyad member, adapted following Formula 2 in McNeish (2018).

Usage

```
getDyReliability(dvn, fit)
```

Arguments

dvn	input dvn list from scrapeVarCross
fit	outputted dyadic cfa lavaan object based on the default (i.e., "configural") dyadCFA() function

Value

a tibble/data frame with calculated omega total coefficients for dyad Member 1 and Member 2

See Also

Other supplemental model calculators: `getDydmacs()`, `getIndistFit()`

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2")
sat.indist.script <-  scriptCFA(dvn, lvname = "Sat")
sat.indist.mod <- lavaan::cfa(sat.indist.script, data = commitmentQ, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = TRUE)
getDyReliability(dvn, sat.indist.mod)
```

---

getIndistFit *A Function that Computes Corrected Fit Indexes According to the ISAT and INULL Models of Olsen & Kenny (2006)*

Description

This function takes the outputted model fit using scriptCFA() with model = "indist", as well as scriptISAT(), and scriptINULL() and computes corrected model fit indexes according to the approach outlined by Olsen & Kenny (2006)

Usage

```
getIndistFit(indmodel, isatmod, inullmod)
```

Arguments

indmodel	input lavaan model object fitted using dyadCFA(model = "indistinguishable")
isatmod	input lavaan model object fitted using ISAT()
inullmod	input lavaan model object fitted using INULL()

Value

A data frame of the original and corrected chi sq, df, p, rmsea, and tli

See Also

Other supplemental model calculators: `getDyReliability()`, `getDydmacs()`

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2")

sat.indist.script <-  scriptCFA(dvn, lvname = "Sat")
sat.indist.mod <- lavaan::cfa(sat.indist.script, data = commitmentQ, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = TRUE)

sat.isat.script <- scriptISAT(dvn, lvxname = "Sat")
sat.isat.mod <- lavaan::cfa(sat.isat.script, data = commitmentQ, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = FALSE)

sat.inull.script <- scriptINULL(dvn, lvxname = "Sat")
sat.inull.mod <- lavaan::cfa(sat.inull.script, data = commitmentQ, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = FALSE)

getIndistFit(sat.indist.mod, sat.isat.mod, sat.inull.mod)
```

---

getInvarCompTable *Compare model fit of nested dyadic invariance models in order from most parsimonious (residual) to least parsimonious (configural)*

Description

Compare model fit of nested dyadic invariance models in order from most parsimonious (residual) to least parsimonious (configural)

Usage

```
getInvarCompTable(mods)
```

Arguments

mods	a list of neted lavaan dyadic invariance models, in the order of residual, intercept, loading, configural

Value

a data frame of model fit statistics for each model, as well as the difference in fit statistics between each model and the previous model

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi",
x_stem = "sat.g", x_delim1 = ".", x_delim2="_", distinguish_1="1", distinguish_2="2")

sat.residual.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = c("loadings", "intercepts", "residuals"), constr_dy_struct = "none")

sat.intercept.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = c("loadings", "intercepts"), constr_dy_struct = "none")

sat.loading.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = c("loadings"), constr_dy_struct = "none")

sat.config.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = "none", constr_dy_struct = "none")

sat.residual.fit <- lavaan::cfa(sat.residual.script, data = commitmentQ,
std.lv = FALSE, auto.fix.first= FALSE, meanstructure = TRUE)

sat.intercept.fit <- lavaan::cfa(sat.intercept.script, data = commitmentQ,
std.lv = FALSE, auto.fix.first= FALSE, meanstructure = TRUE)

sat.loading.fit <- lavaan::cfa(sat.loading.script, data = commitmentQ,
std.lv = FALSE, auto.fix.first= FALSE, meanstructure = TRUE)

sat.config.fit <- lavaan::cfa(sat.config.script, data = commitmentQ,
std.lv = FALSE, auto.fix.first= FALSE, meanstructure = TRUE)

mods <- list(sat.residual.fit, sat.intercept.fit, sat.loading.fit, sat.config.fit)

getInvarCompTable(mods)
```

---

outputModel *A Function That Exports Tables and/or SEM Diagrams based on dySEM models*

Description

This function takes the model from fitted dySEM() scripts and exports table(s) and/or a path diagram figure of expected output.

Usage

```
outputModel(
  dvn,
  model = NULL,
  fit,
  table = TRUE,
```

```
  tabletype = NULL,
  figure = TRUE,
  figtype = NULL,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
model	character input specifying type of model to output (e.g., "cfa", "apim", "cfm")
fit	input object from fitted lavaan model
table	logical input of whether table output is desired. Default is TRUE
tabletype	character input of what type of table(s) is(are) desired. options are "measurement" (i.e,, loadings, intercepts,), "structural" (i.e., latent slopes, such as actor/partner effects, k parameters), or "both" (i.e., both measurement and structural tables)
figure	logical input of whether figure output is desired. Default is TRUE
figtype	character input of what type of figure is desired
writeTo	A character string specifying a directory path to where the file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL (which will throw an error), and examples use a temporary directory created by tempdir().
fileName	A character string specifying a desired base name for the output file(s). If a `fileName` not provided (i.e., default fileName = NULL), then defaults will be used (e.g., "dySEM_table"/"dySEM_table_Measurement"/"dySEM_table_Structural for tables; "dySEM_figure" for figures). The specified name will be automatically appended with the appropriate file extension (i.e., .rtf for tables; .png for figures).

Details

If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

Ignore console (prints unnecessary semPlot::semPaths details). More importantly, prints word files for the table(s) and/or figure, outputted to the users working directory

Examples

```
dvnx <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2")

sat.config.script <- scriptCFA(dvnx, lvname = "Sat", constr_dy_meas = "none",
constr_dy_struct = "none")
```

```
sat.config.mod <- lavaan::cfa(sat.config.script, data = commitmentQ, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = TRUE)

outputModel(dvnx, model = "cfa", fit = sat.config.mod, table = TRUE,
tabletype = "measurement", figure = "TRUE", figtype = "standardized",
writeTo = tempdir(), fileName = "dCFA_configural")
dvnxy <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2",
y_order="spi", y_stem="com", y_delim1 = ".", y_delim2="_")

apim.indist.script <-  scriptAPIM(dvnxy, lvxname = "Sat", lvyname = "Com", est_k = TRUE)

apim.indist.mod <- lavaan::cfa(apim.indist.script, data = commitmentQ, std.lv = FALSE,
auto.fix.first= FALSE, meanstructure = TRUE)

outputModel(dvnxy, model = "apim", fit = apim.indist.mod, table = TRUE,
tabletype = "measurement", figure = "TRUE", figtype = "standardized",
writeTo = tempdir(), fileName = "APIM_indist")
```

`scrapeVarCross` *A Variable Name-Scraping and Indexing Function for cross-sectional data*

Description

This function scrapes the names of indicator variables in a wide-format data set used for dyadic analyses of two latent variables (LV; X and Y), and indexes which indicators correspond to which partner, for which LV. It is used primarily to guide the syntax-writing of the other dySEM functions.

Usage

```
scrapeVarCross(
  dat,
  x_order = "spi",
  x_stem,
  x_delim1 = NULL,
  x_delim2 = NULL,
  x_item_num = "\\d+",
  distinguish_1 = "1",
  distinguish_2 = "2",
  y_order = NULL,
  y_stem = NULL,
  y_delim1 = NULL,
  y_delim2 = NULL,
  y_item_num = "\\d+",
  covs_order = NULL,
  covs_stem = NULL,
  covs_delim1 = NULL,
  covs_delim2 = NULL
)
```

Arguments

dat	input data frame of indicators of a particular LV
x_order	input character for order of (S)tem, (P)artner number, and (I)tem number when creating variable names. Defaults to "spi" (Qualtrics-friendly)
x_stem	input character stem of indicator variables for LV X
x_delim1	optional character to separate stem from partner number (spi) or item number (sip)
x_delim2	optional character to separate stem/partner number (spi) or stem/item number (sip) from from final element of variable name
x_item_num	defaults to scrape all items that match the stem with any digits that follow. Will be updated to allow particular range of values, to make more sub-scale friendly.
distinguish_1	input character used as the identifier for the first partner
distinguish_2	input character used as the identifier for the first partner
y_order	optional character for order of (S)tem, (P)artner number, and (I)tem number when creating variable names. Defaults to "spi" (Qualtrics-friendly). This and other Y-arguments only necessary if there is a latent Y variable to model
y_stem	optional input character stem of indicator variables for LV X
y_delim1	optional character to separate stem from partner number (spi) or item number (sip)
y_delim2	optional character to separate stem/partner number (spi) or stem/item number (sip) from from final element of variable name
y_item_num	defaults to scrape all items that match the stem with any digits that follow. Will be updated to allow particular range of values, to make more sub-scale friendly.
covs_order	optional character for order of (S)tem, (P)artner number, and (I)tem number for any covariate(s). Defaults to NULL. This and other covariate arguments only necessary if there are covariates to be scripted in your model(s).
covs_stem	optional input character stem(s) of indicator variables for covariate(s). Can accept a single stem (e.g., "anx"), or a vector of stems (e.g., c("anx", "dep")). Defaults to NULL.
covs_delim1	optional character to separate stem from partner number (spi) or item number (sip) for covariate(s). Defaults to NULL.
covs_delim2	optional character to separate stem/partner number (spi) or stem/item number (sip) from

Value

a list, referred in short-hand as a "dvn" (dyad variable names list) containing variable names for p1, p2, # of items per LV, characters distinguishing partners, and total number of indicators

Examples

```
dvnx <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2")
dvnxy <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2",
y_order="spi", y_stem="com", y_delim1 = ".", y_delim2="_")
```

scriptAPIM *A Function That Writes, Saves, and Exports Syntax for Fitting Latent Actor-Partner Interdependence Models (APIMs)*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for specifying Actor-Partner Interdependence Models (APIMs). Users can also invoke configural, loading, and/or intercept invariant measurement models, and particular types of structural comparisons.

Usage

```
scriptAPIM(
  dvn,
  scaleset = "FF",
  lvxname,
  lvyname,
  constr_dy_x_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_x_struct = c("variances", "means"),
  constr_dy_y_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_y_struct = c("variances", "means"),
  constr_dy_xy_struct = c("actors", "partners"),
  model = lifecycle::deprecated(),
  equate = lifecycle::deprecated(),
  est_k = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
scaleset	input character to specify how to set the scale of the latent variable(s). Default is "FF" (fixed-factor; see Details for rationale), but user can specify "MV" (Marker Variable)
lvxname	input character to (arbitrarily) name LV X in lavaan syntax
lvyname	input character to (arbitrarily) name LV Y in lavaan syntax
constr_dy_x_meas	input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model

`constr_dy_x_struct`
: input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".

`constr_dy_y_meas`
: input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model

`constr_dy_y_struct`
: input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".

`constr_dy_xy_struct`
: input character vector detailing which structural model parameters to constrain for modeling the predictive association(s) between partners' latent x and y. Default is c("actors", "partners"), but users can also specify "all", "actors_zero", "partners_zero", or "none".

`model`
: Deprecated input character used to specify which level of invariance is modeled. Users should rely upon constr_dy_x_meas/constr_dy_y_meas and constr_dy_x_struct/constr_dy_y_struct instead, for making constraints to the measurement and/or structural portions of the model for latent x and y.

`equate`
: Deprecated input character to specify which type of structural parameters are constrained to equivalency between partners. Users should rely upon constr_dy_xy_struct for making constraints to the structural portion of the model for associative relationship between latent x and y.

`est_k`
: input logical for whether Kenny & Ledermann's (2010) k parameter should be calculated to characterize the dyadic pattern in the APIM. Defaults FALSE, and requires at least a loading-invariant model to be specified, otherwise a warning is returned.

`writeTo`
: A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().

`fileName`
: A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions. Users will receive message if structural comparisons are specified when the recommended level of invariance is not also specified. If user supplies dvn with containing X or Y variables, they are alerted to respecify the dvn object.

See Also

scrapeVarCross which this function relies on

Other script-writing functions: scriptBiDy(), scriptCFA(), scriptCFM(), scriptDyEFA(), scriptINULL(), scriptISAT(), scriptMIM()

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2",
y_order="spi", y_stem="com", y_delim1 = ".", y_delim2="_")
apim.script.indist <-  scriptAPIM(dvn, lvxname = "Sat", lvyname = "Com", est_k = TRUE,
writeTo = tempdir(),
fileName = "latAPIM_indist")
```

---

scriptBiDy *A Function That Writes, Saves, and Exports Syntax for Fitting Bifactor Dyadic (BiDy) models*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for specifying dyadic configural, loading, and intercept invariant BiDy CFA (BiDy-C) or SEM (BiDy-S) Model. Currently only uses fixed-factor scale-setting

Usage

```
scriptBiDy(
  dvn,
  type = "CFA",
  lvxname,
  lvyname,
  constr_dy_x_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_x_struct = c("variances", "means"),
  constr_dy_y_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_y_struct = c("variances", "means"),
  constr_dy_xy_struct = c("actors"),
  model = lifecycle::deprecated(),
  equate = lifecycle::deprecated(),
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
type	input character to specify whether to script a BiDy-CFA ("CFA", default) or BiDy-SEM ("SEM") model

`lvxname` input character to (arbitrarily) name LV X in lavaan syntax

`lvyname` input character to (arbitrarily) name LV Y in lavaan syntax

`constr_dy_x_meas`
input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model. Users may also specify more boutique patterns of bifactor loading constraints with "loadings_source" or "loadings_mutual".

`constr_dy_x_struct`
input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".

`constr_dy_y_meas`
input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model. Users may also specify more boutique patterns of bifactor loading constraints with "loadings_source" or "loadings_mutual".

`constr_dy_y_struct`
input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".

`constr_dy_xy_struct`
input character vector detailing which structural model parameters to constrain for modeling the predictive association(s) between partners' latent x and y. Default is c("actors"), but users can also specify "dyadic_zero" or "none".

`model` Deprecated input character used to specify which level of invariance is modeled. Users should rely upon constr_dy_x_meas/constr_dy_y_meas and constr_dy_x_struct/constr_dy_y_struct instead, for making constraints to the measurement and/or structural portions of the model for latent x and y.

`equate` Deprecated input character to specify which type of structural parameters are constrained to equivalency between partners. Users should rely upon constr_dy_xy_struct for making constraints to the structural portion of the model for associative relationship between latent x and y.

`writeTo` A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().

`fileName` A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions

See Also

Other script-writing functions: scriptAPIM(), scriptCFA(), scriptCFM(), scriptDyEFA(), scriptINULL(), scriptISAT(), scriptMIM()

Examples

```
dvn <- scrapeVarCross(DRES, x_order = "sip", x_stem = "sexsat",
x_delim2=".", distinguish_1="1", distinguish_2="2")

sexsat.bidyc.script <- scriptBiDy(dvn, lvxname = "SexSat", type = "CFA",
writeTo = tempdir(),
fileName = "BiDy_C")

dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2",
y_order="spi", y_stem="com", y_delim1 = ".", y_delim2="_")

comsat.bidys.config.script <- scriptBiDy(dvn, lvxname = "Sat",
lvyname = "Com", type = "SEM",
writeTo = tempdir(),
fileName = "BiDy_S")
```

---

scriptCFA *A Function That Writes, Saves, and Exports Syntax for Fitting Latent Dyadic Confirmatory Factor Analysis (CFA) Models*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for specifying dyadic configural, loading, and intercept invariant measurement models for either a specified X or Y factor.

Usage

```
scriptCFA(
  dvn,
  scaleset = "FF",
  lvname = "X",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"),
  model = lifecycle::deprecated(),
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
scaleset	input character to specify how to set the scale of the latent variable(s). Default is "FF" (fixed-factor; see Details for rationale), but user can specify "MV" (Marker Variable)
lvname	input character to (arbitrarily) name LV in lavaan syntax
constr_dy_meas	input character vector detailing which measurement model parameters to constrain across dyad members. Default is c("loadings", "intercepts", "residuals")(in combination with defaults for constr_dy_struct, an indistinguishable dyadic CFA), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model
constr_dy_struct	input character vector detailing which structural model parameters to constrain across dyad members. Default is c("variances", "means")(in combination with defaults for constr_dy_meas, an indistinguishable dyadic CFA), but user can specify any combination of "variances" and "means", or "none".
model	Depreceated input character used to specify which level of invariance is modeled ("configural", "loading", "intercept", "residual", or "indist"). Users should rely upon constr_dy_meas and constr_dy_struct instead, for making constraints to the measurement and/or structural portions of the model.
writeTo	A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().
fileName	A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Details

By default, many dySEM:: functions (including scriptCFA() default to a fixed-factor method of scale-setting, whereby the latent variance of a given factor is constrained to 1 for both partners in the configurally invariant #model, and then one of these variances is freely estimated in subsequent #models of the invariance testing sequence. We have selected this default for two reasons: (1) the selection of a marker-variable is usually arbitrary, yet can have a large influence on the estimation and testing of of structural parameters (see https://stats.stackexchange.com/questions/402133/in-cfa-does-it-matter-which-factor-loading-is-set-to-1/402732#402732); and (2) the selection of a non-invariant marker-variable can have disastrous down-stream consequences for the identification of non-invariant measurement parameters, following a the rejection of an omnibus #invariance constraint set (see Lee, Preacher, & Little, 2011).

Value

character object of lavaan script that can be passed immediately to lavaan functions

See Also

scrapeVarCross which this function relies on

Other script-writing functions: scriptAPIM(), scriptBiDy(), scriptCFM(), scriptDyEFA(), scriptINULL(), scriptISAT(), scriptMIM()

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2")

sat.indist.script <- scriptCFA(dvn, lvname = "Sat")

sat.lvars.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = "loadings",
constr_dy_struct = "variances")

sat.resids.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = c("loadings", "intercepts", "residuals"),
constr_dy_struct = "none",
writeTo = tempdir(),
fileName = "dCFA_residual")

sat.ints.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = c("loadings", "intercepts"),
constr_dy_struct = "none",
writeTo = tempdir(),
fileName = "dCFA_intercept")

sat.loads.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = c("loadings"),
constr_dy_struct = "none",
writeTo = tempdir(),
fileName = "dCFA_loading")

sat.config.script <- scriptCFA(dvn, lvname = "Sat",
constr_dy_meas = "none",
constr_dy_struct = "none",
writeTo = tempdir(),
fileName = "dCFA_configural")
```

---

scriptCFM *A Function That Writes, Saves, and Exports Syntax for Fitting Latent Common Fate Models (CFMs)*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for specifying Common Fate Models (CFMs). Users can also invoke configural, loading, and/or intercept invariant measurement models, and particular types of structural comparisons.

Usage

```
scriptCFM(
  dvn,
  scaleset = "FF",
  lvxname,
  lvyname,
  constr_dy_x_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_x_struct = c("variances", "means"),
  constr_dy_y_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_y_struct = c("variances", "means"),
  constr_dy_xy_struct = "none",
  model = lifecycle::deprecated(),
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
scaleset	input character to specify how to set the scale of the latent variable(s). Default is "FF" (fixed-factor; see Details for rationale), but user can specify "MV" (Marker Variable)
lvxname	input character to (arbitrarily) name LV X in lavaan syntax
lvyname	input character to (arbitrarily) name LV Y in lavaan syntax
constr_dy_x_meas	input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model
constr_dy_x_struct	input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".
constr_dy_y_meas	input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model
constr_dy_y_struct	input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".
constr_dy_xy_struct	input character vector detailing which structural model parameters to constrain for modeling the predictive association(s) between partners' latent x and y. De-

faults to "none". Options include "p1_zero" or "p2_zero" (to constrain within-person latent residual covariances between X and Y to zero), or "covar_zero" (to constrain both within-person latent residual correlations to zero), and/or "dyadic_zero" (to constrain the dyadic effect to zero).

`model` Deprecated input character used to specify which level of invariance is modeled. Users should rely upon constr_dy_x_meas/constr_dy_y_meas and constr_dy_x_struct/constr_dy_y_struct instead, for making constraints to the measurement and/or structural portions of the model for latent x and y.

`writeTo` A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().

`fileName` A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions. Users will receive message if structural comparisons are specified when the recommended level of invariance is not also specified. If user supplies dvn with containing X or Y variables, they are alerted to respecify the dvn object.

See Also

`scrapeVarCross` which this function relies on

Other script-writing functions: `scriptAPIM()`, `scriptBiDy()`, `scriptCFA()`, `scriptDyEFA()`, `scriptINULL()`, `scriptISAT()`, `scriptMIM()`

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2",
y_order="spi", y_stem="com", y_delim1 = ".", y_delim2="_")
cfm.script.indist <-  scriptCFM(dvn, lvxname = "Sat", lvyname = "Com",
writeTo = tempdir(),
fileName = "CFM_indist")
```

---

scriptDyEFA *A Function That Writes, Saves, and Exports Syntax for Fitting Dyadic Exploratory Factor Analysis (DEFA) Models*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for specifying a dyadic EFA model of a given number of exploratory factors.

Usage

```
scriptDyEFA(
  dvn,
  nFactor = 1,
  constr_dy_meas = "none",
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
nFactor	numeric argument for number of exploratory factors to extract. Defaults to 1. Note that higher values may cause estimation problems as solution becomes over-factored and/or in the presence of insufficient data.
constr_dy_meas	input character vector detailing which measurement model parameters to constrain across dyad members. Default is "none" but user can specify "loadings" and/or "residuals", to fit an exploratory model with loadings and/or residuals constrained across partners
writeTo	A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().
fileName	A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions

See Also

scrapeVarCross which this function relies on

Other script-writing functions: scriptAPIM(), scriptBiDy(), scriptCFA(), scriptCFM(), scriptINULL(), scriptISAT(), scriptMIM()

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2")

sat.defa1.script <- scriptDyEFA(dvn, nFactor = 1,
writeTo = tempdir(), fileName = "DEFA_1fac")
```

scriptINULL *A Function That Writes, Saves, and Exports Syntax for Fitting the I-NULL model for indistinguishable dyads*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for the I-NULL model described in Olsen & Kenny (2006)

Usage

```
scriptINULL(
  dvn,
  lvxname = "X",
  lvyname = NULL,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
lvxname	input character to (arbitrarily) name X LV in lavaan syntax
lvyname	(optional) input character to (arbitrarily) name Y LV in lavaan syntax
writeTo	A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().
fileName	A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions

See Also

scrapeVarCross which this function relies on

Other script-writing functions: scriptAPIM(), scriptBiDy(), scriptCFA(), scriptCFM(), scriptDyEFA(), scriptISAT(), scriptMIM()

Examples

```
dvn <- scrapeVarCross(dat = DRES, x_order = "sip", x_stem = "PRQC", x_delim1 = "_",
x_delim2=".", x_item_num="\\d+", distinguish_1="1", distinguish_2="2")
qual.inull.script <- scriptINULL(dvn, lvxname = "Qual",
writeTo = tempdir(),
fileName = "I-NULL_script")
```

`scriptISAT` *A Function That Writes, Saves, and Exports Syntax for Fitting the I-SAT model for indistinguishable dyads*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for the I-SAT model described in Olsen & Kenny (2006)

Usage

```
scriptISAT(dvn, lvxname = "X", lvyname = NULL, writeTo = NULL, fileName = NULL)
```

Arguments

`dvn`	input dvn list from scrapeVarCross
`lvxname`	input character to (arbitrarily) name X LV in lavaan syntax
`lvyname`	(optional) input character to (arbitrarily) name X LV in lavaan syntax
`writeTo`	A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().
`fileName`	A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions

See Also

`scrapeVarCross` which this function relies on

Other script-writing functions: `scriptAPIM()`, `scriptBiDy()`, `scriptCFA()`, `scriptCFM()`, `scriptDyEFA()`, `scriptINULL()`, `scriptMIM()`

Examples

```
dvn <- scrapeVarCross(dat = DRES, x_order = "sip", x_stem = "PRQC", x_delim1 = "_",
x_delim2=".", x_item_num="\\d+", distinguish_1="1", distinguish_2="2")

qual.isat.script <- scriptISAT(dvn, lvxname = "Qual",
writeTo = tempdir(),
fileName = "I-SAT_script")
```

`scriptMIM` *A Function That Writes, Saves, and Exports Syntax for Fitting Latent Mutual influence Model*

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan() syntax for specifying Mutual Influence Models (MIMs). Users can also invoke configural, loading, and/or intercept invariant measurement models, and particular types of structural comparisons.

Usage

```
scriptMIM(
  dvn,
  scaleset = "FF",
  lvxname,
  lvyname,
  constr_dy_x_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_x_struct = c("variances", "means"),
  constr_dy_y_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_y_struct = c("variances", "means"),
  constr_dy_xy_struct = c("actors", "partners"),
  model = lifecycle::deprecated(),
  equate = lifecycle::deprecated(),
  est_k = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

`dvn`	input dvn list from scrapeVarCross
`scaleset`	input character to specify how to set the scale of the latent variable(s). Default is "FF" (fixed-factor; see Details for rationale), but user can specify "MV" (Marker Variable)
`lvxname`	input character to (arbitrarily) name LV X in lavaan syntax
`lvyname`	input character to (arbitrarily) name LV Y in lavaan syntax

`constr_dy_x_meas`
: input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model

`constr_dy_x_struct`
: input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".

`constr_dy_y_meas`
: input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is c("loadings", "intercepts", "residuals"), but user can specify any combination of "loadings", "intercepts", and "residuals", #or "none" to specify an otherwise unconstrained dyadic configural invariance model

`constr_dy_y_struct`
: input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is c("variances", "means"), but user can specify any combination of "variances" and "means", or "none".

`constr_dy_xy_struct`
: input character vector detailing which structural model parameters to constrain for modeling the predictive association(s) between partners' latent x and y. Default is c("actors", "partners"), but users can also specify "all", "actors_zero", "partners_zero", or "none".

`model`
: Deprecated input character used to specify which level of invariance is modeled. Users should rely upon constr_dy_x_meas/constr_dy_y_meas and constr_dy_x_struct/constr_dy_y_struct instead, for making constraints to the measurement and/or structural portions of the model for latent x and y.

`equate`
: Deprecated input character to specify which type of structural parameters are constrained to equivalency between partners. Users should rely upon constr_dy_xy_struct for making constraints to the structural portion of the model for associative relationship between latent x and y.

`est_k`
: input logical for whether Kenny & Ledermann's (2010) k parameter should be calculated to characterize the dyadic pattern in the APIM. Defaults FALSE, and requires at least a loading-invariant model to be specified, otherwise a warning is returned.

`writeTo`
: A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().

`fileName`
: A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions. Users will receive message if structural comparisons are specified when the recommended level of invariance is not also specified. If user supplies dvn with containing X or Y variables, they are alerted to respecify the dvn object.

See Also

scrapeVarCross which this function relies on

Other script-writing functions: scriptAPIM(), scriptBiDy(), scriptCFA(), scriptCFM(), scriptDyEFA(), scriptINULL(), scriptISAT()

Examples

```
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
x_delim2="_", distinguish_1="1", distinguish_2="2",
y_order="spi", y_stem="com", y_delim1 = ".", y_delim2="_")

mim.script.indist <- scriptMIM(dvn, lvxname = "Sat", lvyname = "Com", est_k = TRUE,
writeTo = tempdir(),
fileName = "MIM_indist")
```

---

scriptObsAPIM *A Function That Writes, Saves, and Exports Syntax for Fitting Observed Actor-Partner Interdependence Models*

Description

A Function That Writes, Saves, and Exports Syntax for Fitting Observed Actor-Partner Interdependence Models

Usage

```
scriptObsAPIM(
  X1 = NULL,
  Y1 = NULL,
  X2 = NULL,
  Y2 = NULL,
  equate = "none",
  k = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

X1	character of vector name containing X variable/composite for partner 1
Y1	character of vector name containing Y variable/composite for partner 1
X2	character of vector name containing X variable/composite for partner 2
Y2	character of vector name containing Y variable/composite for partner 2
equate	character of what parameter(s) to constrain ("actor", "partner", "all"); default is "none" (all freely estimated)
k	input logical for whether Kenny & Ledermann's (2010) k parameter should be calculated to characterize the dyadic pattern in the APIM. Default to FALSE
writeTo	A character string specifying a directory path to where a .txt file of the resulting lavaan script should be written. If set to ".", the .txt file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir().
fileName	A character string specifying a desired base name for the .txt output file. The default is NULL. The specified name will be automatically appended with the .txt file extension. If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

character object of lavaan script that can be passed immediately to lavaan functions.

Examples

```
obsAPIMScript <- scriptObsAPIM (X1 = "SexSatA", Y1 = "RelSatA",
X2 = "SexSatB", Y2 = "RelSatB",
equate = "none",
writeTo = tempdir(),
fileName = "obsAPIM_script")
```

Index