

Package: dySEM (via r-universe)

October 10, 2026

Title Dyadic Structural Equation Modeling

Version 2.1.0

Description Scripting of structural equation models via 'lavaan' for Dyadic Data Analysis, and helper functions for supplemental calculations, tabling, and model visualization.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

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

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

Suggests bslib (>= 0.5.0), janitor, knitr, rmarkdown, shiny (>= 1.7.0), shinyjs, testthat (>= 3.0.0)

VignetteBuilder knitr

LazyData true

Depends R (>= 4.1)

Config/testthat/edition 3

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libnode-dev

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

Date/Publication 2026-05-25 17:44:05 UTC

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

RemoteRef HEAD

RemoteSha 02d871551d163ecb410e76327d49f0a2b8cbfcbb

Contents

build_dvn	3
commitmentM	4
commitmentQ	5
DRES	7
getConstraintTests	8
getDydmacs	9
getDyReliability	10
getIndistFit	10
getInvarCompTable	12
imsM	13
launchScripter	15
outputConstraintTab	16
outputInvarCompTab	17
outputInvarItem	19
outputModel	21
outputParamFig	23
outputParamTab	25
outputUniConstructComp	27
pnrqM	30
prqcQ	31
scrapeVarCross	32
scriptAPIM	35
scriptBiDy	37
scriptBifac	40
scriptCFA	43
scriptCFM	47
scriptCor	49
scriptDyEFA	52
scriptHier	53
scriptINULL	57
scriptISAT	58
scriptMIM	59
scriptObsAPIM	61
scriptPopAPIM	63
scriptPopCor	64
scriptPopUni	65
scriptTwoCross	66
scriptUni	68

build_dvn

*Build a Dyad Variable Names (dvn) List Without a Data Frame***Description**

Creates a dvn structure for use with dySEM population scripters and simulation workflows. Variable names are generated directly from naming conventions without requiring an actual data frame. Compatible with scrapeVarCross naming patterns.

Usage

```
build_dvn(
  n_items_x,
  x_order = "sip",
  x_stem = "x",
  x_delim1 = "",
  x_delim2 = "_",
  distinguish_1 = "A",
  distinguish_2 = "B",
  n_items_y = NULL,
  y_order = NULL,
  y_stem = NULL,
  y_delim1 = NULL,
  y_delim2 = NULL
)
```

Arguments

n_items_x	Number of indicators per partner (X construct; future Y support).
x_order	Order of (S)tem, (I)tem, (P)artner in variable names. One of "sip", "spi", "psi". Default "sip" produces x1_A, x2_A, etc.
x_stem	Character stem of indicator variables (e.g., "x").
x_delim1	Character separating stem from item or partner. Use "" for none.
x_delim2	Character separating item from partner (sip) or partner from item (spi).
distinguish_1	Character identifying the first partner (e.g., "A").
distinguish_2	Character identifying the second partner (e.g., "B").
n_items_y	Optional number of Y indicators per partner. When non-NULL, p1yvarnames, p2yvarnames, and yindper are included.
y_order, y_stem, y_delim1, y_delim2	Y naming (default NULL mirrors X).

Value

A list compatible with dySEM dvn structure:

- p1xvarnames Character vector of variable names for partner 1
- p2xvarnames Character vector of variable names for partner 2
- xindper Number of indicators per partner
- dist1 First partner distinguisher
- dist2 Second partner distinguisher
- indnum Total number of indicators
- p1yvarnames, p2yvarnames, yindper
 Included when n_items_y is set

See Also

Other variable-scraping functions: [scrapeVarCross\(\)](#)

Examples

```
dvn <- build_dvn(  
  n_items_x = 3,  
  x_order = "sip",  
  x_stem = "x",  
  x_delim1 = "",  
  x_delim2 = "_",  
  distinguish_1 = "A",  
  distinguish_2 = "B"  
)  
# Produces x1_A, x2_A, x3_A and x1_B, x2_B, x3_B
```

commitmentM	<i>Ratings of relational satisfaction and commitment from 282 (M)ixed-sex couples</i>
-------------	---

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, 2020)

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

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 <- scriptCor(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	<i>Calculates dmacs difference in expected indicator scores for between dyad members</i>
------------	--

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

For single LV models: a matrix with one column "dMACS" containing d_macs values. For multiple LV models: a named list of matrices, where each element corresponds to one LV and contains a matrix with one column "dMACS" containing d_macs values for that LV.

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 <- scriptCor(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 scriptCor() 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") dyad-CFA() 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 <- scriptCor(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 scriptCor() 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 <- scriptCor(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	<i>Compare model fit of nested dyadic invariance models in order from most parsimonious (residual) to least parsimonious (configural)</i>
-------------------	---

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 <- scriptCor(dvn,
  lvname = "Sat",
  constr_dy_meas = c("loadings", "intercepts", "residuals"), constr_dy_struct = "none"
)

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

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

sat.config.script <- scriptCor(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)

```

imsM

Ratings on the full Investment Model Scale (IMS) from 282 (Mixed-sex couples)

Description

A data set containing 5 ratings for each of (1) satisfaction, (2) quality of alternatives, (3) investment, and 4 (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(imsM)
```

Format

A data frame with 282 rows and 40 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
qalt.g1_f Quality of alternatives item 1 for female partner
qalt.g2_f Quality of alternatives item 2 for female partner
qalt.g3_f Quality of alternatives item 3 for female partner
qalt.g4_f Quality of alternatives item 4 for female partner
qalt.g5_f Quality of alternatives item 5 for female partner
invest.g1_f Investment item 1 for female partner
invest.g2_f Investment item 2 for female partner
invest.g3_f Investment item 3 for female partner
invest.g4_f Investment item 4 for female partner
invest.g5_f Investment 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
qalt.g1_m Quality of alternatives item 1 for male partner
qalt.g2_m Quality of alternatives item 2 for male partner
qalt.g3_m Quality of alternatives item 3 for male partner
qalt.g4_m Quality of alternatives item 4 for male partner
qalt.g5_m Quality of alternatives item 5 for male partner
invest.g1_m Investment item 1 for male partner
invest.g2_m Investment item 2 for male partner
invest.g3_m Investment item 3 for male partner
invest.g4_m Investment item 4 for male partner
invest.g5_m Investment 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. .

launchScripter

Launch the interactive dySEM GUI

Description

Opens a Shiny application that guides researchers through selecting a dyadic model, configuring invariance constraints and variable naming patterns, previewing a path diagram, and generating the corresponding lavaan syntax.

Usage

```
launchScripter(launch.browser = TRUE)
```

Arguments

`launch.browser` Logical; if TRUE (the default), the app opens in the system's default web browser. Set to FALSE to use the RStudio viewer pane instead.

Details

The app supports uni-construct (unidimensional, correlated, hierarchical, bifactor), bi-construct (L-APIM, Common Fate, Mutual Influence, Two-Construct), and multi-construct (multiple correlated factors) models.

No data upload is required. The app constructs variable names from naming patterns you specify, generates lavaan syntax via dySEM scripters, and provides a path diagram preview, copy-to-clipboard, download, and reproducible R code.

Requires the **shiny**, **bslib**, and **shinyjs** packages (listed in Suggests).

Value

Called for its side effect (launching the app). Returns the value from [runApp](#) invisibly.

Examples

```
## Not run:  
launchScripter()  
  
## End(Not run)
```

outputConstraintTab	<i>Evaluate Invariance Equality Constraints in a specified Dyadic Invariance Model</i>
---------------------	--

Description

outputConstraintTab() is used to perform a score test for relaxing each invariance equality constraint between partners in a given dyadic SEM model.

Usage

```
outputConstraintTab(
  constrainFit,
  filterSig = FALSE,
  gtTab = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

constrainFit	A fitted lavaan model with dyadic invariance equality constraints
filterSig	A logical indicating whether to filter for significant constraints (default is FALSE)
gtTab	A logical input indicating whether to generate the table(s) in gt::gt() table object format (TRUE). By default (FALSE), the table(s) are generated in tibble::tibble() format. Users can also apply the writeTo argument if they wish to export the gt::gt() table object(s).
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Details

- If gtTab = TRUE and writeTo is specified, then output will simultaneously be saved as a .rtf file to the user's specified directory.
- If output file is successfully saved, a confirmation message will be printed to the console.
- If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

A `tibble::tibble()` if `gtTab = FALSE` (default), or `gt::gt()` object if `gtTab = TRUE`, 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 <- scriptCor(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
)

outputConstraintTab(sat.resids.mod,
  filterSig = FALSE,
  gtTab = TRUE, writeTo = tempdir(), fileName = "dCFA_Residual"
)
```

outputInvarCompTab	<i>Compare Model Fit of Nested Dyadic Invariance Models</i>
--------------------	---

Description

`outputInvarCompTab()` is used to compare the model fit of nested dyadic invariance models in order from most parsimonious (residual) to least parsimonious (configural)

Usage

```
outputInvarCompTab(
  mods,
  parsimonyFirst = FALSE,
  gtTab = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

<code>mods</code>	A list of nested lavaan dyadic invariance models, in the order of residual, intercept, loading, configural
<code>parsimonyFirst</code>	A logical input indicating whether to prioritize the residual dyadic invariance (i.e. most parsimonious measurement model) as the baseline model for nested comparisons, or to prioritize the configural dyadic invariance (i.e. least parsimonious measurement model). Defaults to FALSE (i.e., configural dyadic invariance is the baseline model).
<code>gtTab</code>	A logical input indicating whether to generate the table(s) in <code>gt::gt()</code> table object format (TRUE). By default (FALSE), the table(s) are generated in <code>tibble::tibble()</code> format. Users can also apply the <code>writeTo</code> argument if they wish to export the <code>gt::gt()</code> table object(s).
<code>writeTo</code>	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
<code>fileName</code>	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Details

- If `gtTab = TRUE` and `writeTo` is specified, then output will simultaneously be saved as a .rtf file to the user's specified directory.
- If output file is successfully saved, a confirmation message will be printed to the console.
- If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

A `tibble::tibble()` if `gtTab = FALSE` (default), or `gt::gt()` object if `gtTab = TRUE`, 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 <- scriptCor(dvn,
  lvname = "Sat",
  constr_dy_meas = c("loadings", "intercepts", "residuals"), constr_dy_struct = "none"
)
```

```

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

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

sat.config.script <- scriptCor(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)

outputInvarCompTab(mods,
  parsimonyFirst = FALSE,
  gtTab = TRUE, writeTo = tempdir(), fileName = "dcFA_Invar_Standard"
)

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

outputInvarCompTab(mods,
  parsimonyFirst = TRUE,
  gtTab = TRUE, writeTo = tempdir(), fileName = "dcFA_Invar_Reverse"
)

```

Description

outputInvarItem() generates a summary table showing which items exhibit noninvariance in loadings and/or intercepts based on a partial invariance model script, along with dMACS effect sizes for each item.

Usage

```
outputInvarItem(
  dvn,
  fit,
  partialScript,
  dat,
  nodewidth = 0.01,
  lowerLV = -5,
  upperLV = 5,
  gtTab = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	input dvn list from scrapeVarCross
fit	outputted dyadic cfa lavaan object from the partial invariance model
partialScript	character string containing the lavaan syntax for the partial invariance model
dat	data frame of indicators used to fit the model (needed for dMACS calculation)
nodewidth	space between nodes during quadrature approximation for dMACS (default = .01)
lowerLV	lowest latent variable value evaluated for dMACS (default = -5)
upperLV	greatest latent variable value evaluated for dMACS (default = 5)
gtTab	A logical input indicating whether to generate the output in gt::gt() table object format (TRUE). By default (FALSE), the output is generated in tibble::tibble() format. Users can also apply the writeTo argument if they wish to export the gt:gt() table object.
writeTo	A character string specifying a directory path to where the gt::gt() table object should be saved. If set to ".", the file will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output gt::gt() file.

Details

- The function parses the lavaan syntax to determine which loadings and intercepts are constrained to equality across partners (invariant) versus freed (noninvariant).
- LoadingNoninvariance = "Yes" indicates the loading was freed in the partial invariance model.

- InterceptNoninvariance = "Yes" indicates the intercept was freed in the partial invariance model.
- dMACS values are calculated using the getDydmacs() function.
- If gtTab = TRUE and writeTo is specified, then output will simultaneously be saved as a .rtf file to the user's specified directory.
- If output file is successfully saved, a confirmation message will be printed to the console.
- If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

A `tibble::tibble()` if `gtTab = FALSE` (default), or `gt::gt()` object if `gtTab = TRUE`, with columns for Item, LoadingNoninvariance, InterceptNoninvariance, and dMACS. For multi-LV structures, an additional LV column is included; in gt tables, this is used as the row group label with Item names as row labels within each group.

Examples

```
## Not run:
dvn <- scrapeVarCross(dat = commitmentQ, x_order = "spi", x_stem = "sat.g", x_delim1 = ".",
  x_delim2 = "_", distinguish_1 = "1", distinguish_2 = "2")

# Create and fit a partial invariance model
partial.script <- scriptCor(dvn, lvname = "Sat",
  constr_dy_meas = c("loadings", "intercepts"),
  constr_dy_struct = "none")

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

outputInvarItem(dvn, partial.fit, partial.script, dat = commitmentQ,
  gtTab = TRUE, writeTo = tempdir(), fileName = "item_invar")

## End(Not run)
```

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 output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.

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 <- scriptCor(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")
dvnxxy <- 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(dvnxxy, lvxname = "Sat", lvynname = "Com", est_k = TRUE)

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

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

```

outputParamFig

A function That Exports SEM diagrams based on dySEM models

Description

This function takes the model from fitted dySEM scripts and exports .png path diagram figures of expected output.

Usage

```
outputParamFig(fit, figtype = NULL, writeTo = NULL, fileName = NULL)
```

Arguments

- | | |
|---------|---|
| fit | Input object from fitted lavaan model. |
| figtype | A character input of what type of model is desired: <ul style="list-style-type: none"> • "unstandardized" for unstandardized path coefficients. • "standardized" for standardized path coefficients. • "labels" for labelled parameters. |

writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Details

- The function uses `semPlot::semPaths()` to create a qgraph object of the desired SEM path diagram.
- After execution, a `semPlot::semPaths()` message will be printed to the console confirming the directory path of the saved output file.
- If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

A qgraph object of the desired SEM path diagram, which is simultaneously exported as a .png file to the specified 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 <- scriptCor(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
)

outputParamFig(sat.config.mod,
  figtype = "standardized",
  writeTo = tempdir(), fileName = "dCFA_configural"
)

dvnxxy <- 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
)

outputParamFig(apim.indist.mod,
  figtype = "standardized",
  writeTo = tempdir(), fileName = "APIM_indist"
)

```

outputParamTab

A Function That Exports Tables based on dySEM models

Description

This function takes the model from fitted dySEM scripts and creates tables of expected output and/or exports them as .rtfs.

Usage

```

outputParamTab(
  dvn,
  model = NULL,
  fit,
  tabletype = NULL,
  gtTab = FALSE,
  writeTo = NULL,
  fileName = NULL
)

```

Arguments

dvn	Input dvn list from scrapeVarCross()
model	A character input specifying type of model to output: "cfa", "bidyc", "apim", "mim", "cfm", or "bidys".
fit	input object from fitted lavaan model.
tabletype	A character input of what type of table(s) is(are) desired. Options are "measurement" (i.e., loadings, intercepts, etc.), "structural" (i.e., latent slopes, such as actor/partner effects, k parameters), "both" (i.e., both measurement and structural tables), or "correlation" (for a table of factor correlations).
gtTab	A logical input indicating whether to generate the table(s) in gt::gt() table object format (TRUE). By default (FALSE), the table(s) are generated in tibble::tibble() format. Users can also apply the writeTo argument if they wish to export the gt::gt() table object(s).

<code>writeTo</code>	A character string specifying a directory path to where the output file(s) should be saved. If set to <code>"."</code> , the file(s) will be written to the current working directory. The default is <code>NULL</code> , and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
<code>fileName</code>	A character string specifying a desired base name for the output file. The default is <code>NULL</code> . The specified name will be automatically appended with the appropriate file extension (e.g., <code>.txt</code> for lavaan scripts, <code>.rtf</code> for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Details

- If `gtTab = TRUE` and `writeTo` is specified, then output will simultaneously be saved as a `.rtf` file to the user's specified directory.
- If output file(s) is(are) successfully saved, a confirmation message will be printed to the console.
- If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

A `tibble::tibble()` if `gtTab = FALSE` (default), or `gt::gt()` object if `gtTab = TRUE`, of specified model parameter estimates and corresponding statistical tests.

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 <- scriptCor(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
)

outputParamTab(dvnx,
  model = "cfa", sat.config.mod, tabletype = "measurement",
  writeTo = tempdir(), fileName = "dCFA_configural"
)

dvnxxy <- 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
)

outputParamTab(dvnxy,
  model = "cfa", sat.config.mod, tabletype = "measurement",
  writeTo = tempdir(), fileName = "APIM_indist"
)

```

outputUniConstructComp

A Function That Fits and Compares Competing Dyadic Uni-construct Models

Description

This function takes the outputted object from `scrapeVarCross()` along with the corresponding dataset and automatically tests competing uni-construct dyadic models for the latent variable under consideration. It inspects four possible model variants:

- Bifactor (scripted via `dySEM::scriptBifac`)
- Hierarchical (scripted via `dySEM::scriptHier`)
- Correlated Factors (scripted via `dySEM::scriptCor`)
- Unidimensional (scripted via `dySEM::scriptUni`)

Usage

```

outputUniConstructComp(
  dvn,
  dat,
  indexes = c("df", "chisq", "cfi", "rmsea", "bic", "GenTEFI"),
  ...,
  gtTab = FALSE,
  writeTo = NULL,
  fileName = NULL
)

```

Arguments

dvn	Input dvn list from <code>scrapeVarCross()</code> .
dat	Input data frame containing the dataset for model estimation.

indexes	Input character vector specifying which index(es) to return. Default is <code>c("df", "chisq", "cfi", "rmsea", "bic", "GenTEFI")</code> . Note: <ul style="list-style-type: none"> Valid entries include "GenTEFI"—the Generalized Total Entropy Fit Index (see Golino et al., 2024)—and those from <code>lavaan::fitMeasures()</code>. If "chisq" is entered, chi-squared difference tests are automatically performed via <code>lavaan::lavTestLRT()</code>, and the resulting p-values are added to the output.
...	Additional arguments to be passed to <code>lavaan::cfa()</code> , allowing users to customize model estimation settings. By default, the models are fit with maximum-likelihood estimation (<code>estimator = "ml"</code>) and missing data are handled via list-wise deletion (<code>missing = "listwise"</code>), as per <code>lavaan::cfa()</code> 's default behaviour.
gtTab	A logical input indicating whether to generate the table(s) in <code>gt::gt()</code> table object format (TRUE). By default (FALSE), the table(s) are generated in <code>tibble::tibble()</code> format. Users can also apply the <code>writeTo</code> argument if they wish to export the <code>gt::gt()</code> table object(s).
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to <code>."</code> , the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., <code>.txt</code> for lavaan scripts, <code>.rtf</code> for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Details

- If "chisq" is included in `indexes`, the specific form of the applied chi-squared difference test (e.g., standard vs. robust) is determined automatically by `lavaan::lavTestLRT()`, based on the model estimation method used.
- If `gtTab = TRUE` and `writeTo` is specified, then output will simultaneously be saved as a `.rtf` file to the user's specified directory.
- If output file is successfully saved, a confirmation message will be printed to the console.
- If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

A list containing up to two components:

- Indexes: A `tibble::tibble()` if `gtTab = FALSE` (default), or `gt::gt()` object if `gtTab = TRUE`, with the desired index(es) for each fitted model (requested via the `indexes` argument).
- GenTEFI: A `tibble::tibble()` of the GenTEFI (if "GenTEFI" is included in the `indexes` argument).

Examples

```

dvn <- scrapeVarCross(
  commitmentM,
  x_order = "sip",
  x_stem = "sat.g",
  x_delim1 = "",
  x_delim2 = "_",
  distinguish_1 = "f",
  distinguish_2 = "m"
)

# Quick example for CRAN checks
outputUniConstructComp(
  dvn,
  commitmentM,
  indexes = c("df", "bic"),
  missing = "listwise"
)

# More comprehensive examples (slower due to FIML estimation)
outputUniConstructComp(
  dvn,
  commitmentM,
  missing = "fiml"
)

outputUniConstructComp(
  dvn,
  commitmentM,
  indexes = c("df", "bic"),
  missing = "fiml"
)

outputUniConstructComp(
  dvn,
  commitmentM,
  indexes = c("df", "bic"),
  estimator = "ml",
  missing = "fiml"
)

outputUniConstructComp(
  dvn,
  commitmentM,
  indexes = c("df", "bic"),
  missing = "fiml",
  gtTab = TRUE,
  writeTo = tempdir(),
  fileName = "uni-construct-dyad-models"
)

```

pnrqM	<i>Ratings on items from the Positive-Negative Relationship Quality Scale (PNRQ; Rogge et al., 2017) from 219 (M)ixed-sex couples</i>
-------	---

Description

A data set containing ratings on items (4 each) assessing romantic (1) satisfaction and (2) dissatisfaction for each member of a mixed-sex dyad. Positive-Negative Relationship Quality Scale (Rogge et al., 2017). Data are from Prine et al. (Under Review). Variable names follow a stem-item-partner ("sip") order, with a delimiter ("_") between the item number and distinguishing partner character.

Usage

```
data(pnrqM)
```

Format

A data frame with 219 rows and 16 variables. Participants responded—on a six-point scale (1 = "Not at all true", 6 = "Completely true")—to the prompt, "My relationship is...":

sat.pnrq1_w Enjoyable for partner w
sat.pnrq2_w Pleasant for partner w
sat.pnrq3_w Strong for partner w
sat.pnrq4_w Alive for partner w
dsat.pnrq1_w Miserable for partner w
dsat.pnrq2_w Bad for partner w
dsat.pnrq3_w Empty for partner w
dsat.pnrq4_w Lifeless for partner w
sat.pnrq1_m Enjoyable for partner m
sat.pnrq2_m Pleasant for partner m
sat.pnrq3_m Strong for partner m
sat.pnrq4_m Alive for partner m
dsat.pnrq1_m Miserable for partner m
dsat.pnrq2_m Bad for partner m
dsat.pnrq3_m Empty for partner m
dsat.pnrq4_m Lifeless for partner m

References

Prine, M., Sakaluk, J. K., Camanto, O. J., & Quinn-Nilas, C. (Under Review).

prqcQ

Ratings on items from the Perceived Relationship Quality Components (PRQC) Inventory from 118 (Q)ueer couples

Description

A data set containing ratings on items (3 each) assessing romantic: (1) satisfaction, (2) commitment, (3) intimacy, (4) trust, (5) passion, and (6) love for each member of a dyad in which one or more members identify as LGBTQ+. Perceived Relationship Quality Components (PRQC) Inventory (Fletcher, Simpson, & Thomas, 2000). 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(prqcQ)
```

Format

A data frame with 118 rows and 36 variables:

prqc.1_1 Satisfaction item 1 for partner 1
prqc.1_2 Satisfaction item 2 for partner 1
prqc.1_3 Satisfaction item 3 for partner 1
prqc.1_4 Commitment item 1 for partner 1
prqc.1_5 Commitment item 2 for partner 1
prqc.1_6 Commitment item 3 for partner 1
prqc.1_7 Intimacy item 1 for partner 1
prqc.1_8 Intimacy item 2 for partner 1
prqc.1_9 Intimacy item 3 for partner 1
prqc.1_10 Trust item 1 for partner 1
prqc.1_11 Trust item 2 for partner 1
prqc.1_12 Trust item 3 for partner 1
prqc.1_13 Passion item 1 for partner 1
prqc.1_14 Passion item 2 for partner 1
prqc.1_15 Passion item 3 for partner 1
prqc.1_16 Love item 1 for partner 1
prqc.1_17 Love item 2 for partner 1
prqc.1_18 Love item 3 for partner 1
prqc.2_1 Satisfaction item 1 for partner 2

prqc.2_2 Satisfaction item 2 for partner 2
prqc.2_3 Satisfaction item 3 for partner 2
prqc.2_4 Commitment item 1 for partner 2
prqc.2_5 Commitment item 2 for partner 2
prqc.2_6 Commitment item 3 for partner 2
prqc.2_7 Intimacy item 1 for partner 2
prqc.2_8 Intimacy item 2 for partner 2
prqc.2_9 Intimacy item 3 for partner 2
prqc.2_10 Trust item 1 for partner 2
prqc.2_11 Trust item 2 for partner 2
prqc.2_12 Trust item 3 for partner 2
prqc.2_13 Passion item 1 for partner 2
prqc.2_14 Passion item 2 for partner 2
prqc.2_15 Passion item 3 for partner 2
prqc.2_16 Love item 1 for partner 2
prqc.2_17 Love item 2 for partner 2
prqc.2_18 Love item 3 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. .#'

scrapeVarCross	<i>A Variable Name-Scraping and Indexing Function for cross-sectional data</i>
----------------	--

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+",
    var_list = NULL,
    var_list_order = NULL,
    var_list_item_num = "\\d+",
    group = NULL,
    covs_order = NULL,
    covs_stem = NULL,
    covs_delim1 = NULL,
    covs_delim2 = NULL,
    verbose = TRUE
)

```

Arguments

<code>dat</code>	input data frame of indicators of a particular LV
<code>x_order</code>	input character for order of (S)tem, (P)artner number, and (I)tem number when creating variable names. Defaults to "spi" (Qualtrics-friendly), but can alternatively take "sip" or "psi"
<code>x_stem</code>	input character stem of indicator variables for LV X
<code>x_delim1</code>	optional character to separate stem from partner number (spi) or item number (sip)
<code>x_delim2</code>	optional character to separate stem/partner number (spi) or stem/item number (sip) from from final element of variable name
<code>x_item_num</code>	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.
<code>distinguish_1</code>	input character used as the identifier for the first partner
<code>distinguish_2</code>	input character used as the identifier for the first partner
<code>y_order</code>	optional character for order of (S)tem, (P)artner number, and (I)tem number when creating variable names. Defaults to "spi" (Qualtrics-friendly), but can alternatively take "sip" or "psi". This and other Y-arguments only necessary if there is a latent Y variable to model
<code>y_stem</code>	optional input character stem of indicator variables for LV X
<code>y_delim1</code>	optional character to separate stem from partner number (spi) or item number (sip)
<code>y_delim2</code>	optional character to separate stem/partner number (spi) or stem/item number (sip) from from final element of variable name
<code>y_item_num</code>	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.

<code>var_list</code>	optional named list of indicator variable information, if more than one LV is to be scripted (e.g., a dyadic CFA with multiple sub-scales from the same measure). If supplied, this list <i>must</i> contain the following elements: "stem" (a vector of stems), "delim1" (a vector of delimiting characters), and "delim2" (a vector of subsequently delimiting characters). Optionally may include numeric vectors "min_num" and "max_num" if indicators for different LVs share the same stem and must be separated by range of item numbers within a measure.
<code>var_list_order</code>	optional character for order of (S)tem, (P)artner number, and (I)tem number for any of the indicator variables of a multi-LV model (i.e., this functionality assumes the same ordering of elements throughout)
<code>var_list_item_num</code>	optional character for item number of any of the indicator variables of a multi-LV model
<code>group</code>	optional character naming a column in <code>dat</code> that defines groups for multi-group analysis. When supplied, the returned <code>dvn</code> will include <code>group</code> (variable name), <code>group_n</code> (number of levels), and <code>group_levels</code> (values). Required for multi-group scripters when <code>constr_group_meas</code> or <code>constr_group_struct</code> are non-NULL.
<code>covs_order</code>	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).
<code>covs_stem</code>	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.
<code>covs_delim1</code>	optional character to separate stem from partner number (spi) or item number (sip) for covariate(s). Defaults to NULL.
<code>covs_delim2</code>	optional character to separate stem/partner number (spi) or stem/item number (sip) from
<code>verbose</code>	logical indicating whether to print a summary of scraped variables to the console. Defaults to TRUE.

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

See Also

Other variable-scraping functions: [build_dvn\(\)](#)

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"),
  includeMeanStruct = FALSE,
  model = lifecycle::deprecated(),
  equate = lifecycle::deprecated(),
  est_k = FALSE,
  writeTo = NULL,
  fileName = NULL
)

```

Arguments

dvn	input dvn list from <code>scrapeVarCross</code>
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

<code>constr_dy_x_meas</code>	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
<code>constr_dy_x_struct</code>	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", "means", and/or "orthogonal" (zero dyadic covariance between partners' latent X), or "none".
<code>constr_dy_y_meas</code>	input character vector detailing which measurement model parameters to constrain across dyad members for latent Y. 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
<code>constr_dy_y_struct</code>	input character vector detailing which structural model parameters to constrain across dyad members for latent Y. Default is c("variances", "means"), but user can specify any combination of "variances", "means", and/or "orthogonal" (zero dyadic covariance between partners' latent Y), or "none".
<code>constr_dy_xy_struct</code>	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".
<code>includeMeanStruct</code>	input logical for whether the user wants to include the mean structure in the model. Defaults FALSE, to support subsequent calculation of dynamic fit indexes (see Details)
<code>model</code>	Deprecated input character used to specify which level of invariance is modeled. Users should rely upon <code>constr_dy_x_meas/constr_dy_y_meas</code> and <code>constr_dy_x_struct/constr_dy_y_struct</code> instead, for making constraints to the measurement and/or structural portions of the model for latent x and y.
<code>equate</code>	Deprecated input character to specify which type of structural parameters are constrained to equivalency between partners. Users should rely upon <code>constr_dy_xy_struct</code> for making constraints to the structural portion of the model for associative relationship between latent x and y.
<code>est_k</code>	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.
<code>writeTo</code>	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .

fileName A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). 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 bi-construct script-writing functions: [scriptBiDy\(\)](#), [scriptCFM\(\)](#), [scriptMIM\(\)](#), [scriptObsAPIM\(\)](#), [scriptTwoCross\(\)](#)

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", lvynname = "Com", est_k = TRUE,
  writeTo = tempdir(),
  fileName = "latAPIM_indist"
)
```

scriptBiDy	<i>A Function That Writes, Saves, and Exports Syntax for Fitting Bifactor Dyadic (BiDy) models</i>
------------	--

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,
```

```

lvynname,
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
lvynname	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 <code>constr_dy_x_meas/constr_dy_y_meas</code> and <code>constr_dy_x_struct/constr_dy_y_struct</code> 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 <code>constr_dy_xy_struct</code> 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 the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). 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 bi-construct script-writing functions: [scriptAPIM\(\)](#), [scriptCFM\(\)](#), [scriptMIM\(\)](#), [scriptObsAPIM\(\)](#), [scriptTwoCross\(\)](#)

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")
```

scriptBifac

A Function That Writes, Saves, and Exports Syntax for Fitting Bifactor Dyadic Models

Description

This function takes the outputted object from `scrapeVarCross()` and automatically writes, returns, and exports (.txt) lavaan syntax for specifying dyadic configural, loading, intercept, and residual invariant bifactor models.

Usage

```
scriptBifac(
  dvn,
  scaleset = "FF",
  lvname = "X",
  lvar = "X",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"),
  constr_group_meas = NULL,
  constr_group_struct = NULL,
  writeTo = NULL,
  fileName = NULL,
  outputType = "lavaan script"
)
```

Arguments

dvn	Input dvn list from <code>scrapeVarCross()</code>
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 the latent variable in lavaan syntax
lvar	Input character to specify whether the latent variable represents "X" or "Y" in the model. Default is "X". This argument controls parameter labeling and which indicator variables are used from the dvn.
constr_dy_meas	Input character vector detailing which measurement model parameters to constrain across dyad members.
constr_dy_struct	Input character vector detailing which structural model parameters to constrain across dyad members. Default is <code>c("variances", "means")</code> (in combination with defaults for <code>constr_dy_meas</code> , an indistinguishable correlated dyadic factors model), but user can specify any combination of "variances" and "means", or "none".

constr_group_meas	Optional character vector detailing which measurement model parameters to constrain across groups. Valid values: "loadings", "intercepts", "residuals", or "none". Default NULL = single-group. Requires dvn from scrapeVarCross(..., group = "varname").
constr_group_struct	Optional character vector detailing which structural model parameters to constrain across groups. Valid values: "variances", "means", or "none". Default NULL = single-group. Requires dvn from scrapeVarCross(..., group = "varname").
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.
outputType	Character string specifying the type of output to return. Options are "lavaan script" (default) to return a character object of lavaan syntax that can be passed immediately to lavaan functions, or "syntax components" to return a structured list of model components.

Details

- By default, many `dySEM::` functions (including `scriptBifac()`) 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 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 (when `outputType = "lavaan script"`), or a structured list of model components (when `outputType = "syntax components"`).

See Also

[scrapeVarCross](#) which this function relies on.

Other uni-construct script-writing functions: [scriptCor\(\)](#), [scriptHier\(\)](#), [scriptUni\(\)](#)

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 <- scriptBifac(
  dvn,
  scaleset = "FF",
  lvname = "Sat"
)

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

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

sat.ints.script <- scriptBifac(
  dvn,
  scaleset = "FF",
  lvname = "Sat",
  constr_dy_meas = c("loadings", "intercepts"),
  constr_dy_struct = "none",
  writeTo = tempdir(),
  fileName = "dBiFac_intercept"
)

sat.loads.script <- scriptBifac(
  dvn,
  scaleset = "FF",
  lvname = "Sat",
  constr_dy_meas = c("loadings"),
  constr_dy_struct = "none",
  writeTo = tempdir(),

```

```

    fileName = "dBiFac_loading"
  )

  sat.config.script <- scriptBifac(
    dvn,
    scaleset = "FF",
    lvname = "Sat",
    constr_dy_meas = "none",
    constr_dy_struct = "none",
    writeTo = tempdir(),
    fileName = "dBiFac_configural"
  )

  sat.source.script <- scriptBifac(
    dvn,
    scaleset = "FF",
    lvname = "Sat",
    constr_dy_meas = "loadings_source",
    constr_dy_struct = "none",
    writeTo = tempdir(),
    fileName = "dBiFac_source"
  )

```

scriptCFA

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

Description

This function takes the outputted object from `scrapeVarCross()` when the `var_list` argument has been used, and automatically writes, returns, and exports (.txt) lavaan() syntax for specifying dyadic configural, loading, and intercept invariant measurement models for either a group of latent variables (e.g., different sub-scales from a self-report measures).

`scriptMultiCor()` is a thin wrapper around `scriptCFA()` provided as a synonym for users who think of these models as dyadic multi-factor correlated-factor models. It takes the same arguments as `scriptCFA()` and returns the identical lavaan syntax.

Usage

```

scriptCFA(
  dvn,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"),
  writeTo = NULL,
  fileName = NULL
)

```

```

scriptMultiCor(
  dvn,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"),
  writeTo = NULL,
  fileName = NULL
)

```

Arguments

dvn	input dvn list from scrapeVarCross(); scrapeVarCross <i>must</i> have been run with the var_list argument
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).
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".
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). 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 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

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

See Also

[scrapeVarCross](#) which this function relies on

Other multi-construct script-writing functions: [scriptDyEFA\(\)](#)

Examples

```
# When different factor use distinct stems:
imsList <- list(
  lvnames = c("Sat", "Q_Alt", "Invest", "Comm"),
  stem = c("sat.g", "qalt.g", "invest.g", "com"),
  delim1 = c("", "", "", ""),
  delim2 = c("_", "_", "_", "_")
)

dvnIMS <- scrapeVarCross(imsM,
  var_list = imsList,
  var_list_order = "sip",
  distinguish_1 = "f",
  distinguish_2 = "m"
)

script.ims.config <- scriptCFA(dvnIMS,
  scaleset = "FF",
  constr_dy_meas = "none",
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "ims_config"
)

script.ims.load <- scriptCFA(dvnIMS,
  scaleset = "FF",
  constr_dy_meas = c("loadings"),
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "ims_load"
)

script.ims.int <- scriptCFA(dvnIMS,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts"),
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "ims_int"
)
```

```

script.ims.res <- scriptCFA(dvnIMS,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "ims_res"
)

script.ims.indist <- scriptCFA(dvnIMS,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"), writeTo = tempdir(), fileName = "ims_indist"
)

# When different factor use the same stem and muse be detected through numeric position

prqcList <- list(
  lvnames = c("Sat", "Comm", "Intim", "Trust", "Pass", "Love"),
  stem = c("prqc", "prqc", "prqc", "prqc", "prqc", "prqc"),
  delim1 = c(".", ".", ".", ".", ".", "."),
  delim2 = c("-", "-", "-", "-", "-", "-"),
  min_num = c(1, 4, 7, 10, 13, 16),
  max_num = c(3, 6, 9, 12, 15, 18)
)

dvnPRQC <- scrapeVarCross(prqcQ,
  var_list = prqcList,
  var_list_order = "spi",
  distinguish_1 = "1",
  distinguish_2 = "2"
)

script.prqc.config <- scriptCFA(dvnPRQC,
  scaleset = "FF",
  constr_dy_meas = "none",
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "prqc_config"
)

script.prqc.load <- scriptCFA(dvnPRQC,
  scaleset = "FF",
  constr_dy_meas = c("loadings"),
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "prqc_load"
)

script.prqc.int <- scriptCFA(dvnPRQC,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts"),
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "prqc_int"
)

script.prqc.res <- scriptCFA(dvnPRQC,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = "none", writeTo = tempdir(), fileName = "prqc_res"
)

```

```
script.prqc.indist <- scriptCFA(dvnPRQC,
  scaleset = "FF",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"), writeTo = tempdir(), fileName = "prqc_indist"
)
```

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,
  lvynname,
  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 <code>scrapeVarCross</code>
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
lvynname	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. Defaults 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 the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). 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 bi-construct script-writing functions: [scriptAPIM\(\)](#), [scriptBiDy\(\)](#), [scriptMIM\(\)](#), [scriptObsAPIM\(\)](#), [scriptTwoCross\(\)](#)

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")
```

scriptCor

A Function That Writes, Saves, and Exports Syntax for Fitting Correlated Dyadic Factor Models

Description

This function takes the outputted object from `scrapeVarCross()` and automatically writes, returns, and exports (.txt) lavaan syntax for specifying dyadic configural, loading, intercept, and residual invariant two-factor models.

Usage

```
scriptCor(
  dvn,
  scaleset = "FF",
  lvname = "X",
  lvar = "X",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"),
  constr_group_meas = NULL,
  constr_group_struct = NULL,
  writeTo = NULL,
  fileName = NULL,
  outputType = "lavaan script"
)
```

Arguments

dvn	Input dvn list from <code>scrapeVarCross()</code>
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 the latent variable in lavaan syntax
lvar	Input character to specify whether the latent variable represents "X" or "Y" in the model. Default is "X". This argument controls parameter labeling (e.g., lx vs ly for loadings, tx vs ty for intercepts).
constr_dy_meas	Input character vector detailing which measurement model parameters to constrain across dyad members.
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 correlated dyadic factors model), but user can specify any combination of "variances" and "means", or "none".
constr_group_meas	Optional character vector detailing which measurement model parameters to constrain across groups. Valid values: "loadings", "intercepts", "residuals", "residual.covariances", or "none". Default NULL = single-group. Requires dvn from scrapeVarCross(..., group = "varname").
constr_group_struct	Optional character vector detailing which structural model parameters to constrain across groups. Valid values: "variances", "means", or "none". Default NULL = single-group. Requires dvn from scrapeVarCross(..., group = "varname").
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.
outputType	Character string specifying the type of output to return. Options are "lavaan script" (default) to return a character object of lavaan syntax that can be passed immediately to lavaan functions, or "syntax components" to return a structured list of model components.

Details

- By default, many `dySEM::` functions (including `scriptCor()`) 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 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 (when `outputType = "lavaan script"`), or a structured list of model components (when `outputType = "syntax components"`).

See Also

[scrapeVarCross](#) which this function relies on.

Other uni-construct script-writing functions: [scriptBifac\(\)](#), [scriptHier\(\)](#), [scriptUni\(\)](#)

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 <- scriptCor(
  dvn,
  scaleset = "FF",
  lvname = "Sat"
)

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

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

sat.ints.script <- scriptCor(
```

```

    dvn,
    scaleset = "FF",
    lvname = "Sat",
    constr_dy_meas = c("loadings", "intercepts"),
    constr_dy_struct = "none",
    writeTo = tempdir(),
    fileName = "dCor_intercept"
  )

  sat.loads.script <- scriptCor(
    dvn,
    scaleset = "FF",
    lvname = "Sat",
    constr_dy_meas = c("loadings"),
    constr_dy_struct = "none",
    writeTo = tempdir(),
    fileName = "dCor_loading"
  )

  sat.config.script <- scriptCor(
    dvn,
    scaleset = "FF",
    lvname = "Sat",
    constr_dy_meas = "none",
    constr_dy_struct = "none",
    writeTo = tempdir(),
    fileName = "dCor_configural"
  )

```

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 the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). 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 multi-construct script-writing functions: [scriptCFA\(\)](#)

Examples

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

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

scriptHier	<i>A Function That Writes, Saves, and Exports Syntax for Fitting Hierarchical Dyadic Factor Models</i>
------------	--

Description

This function takes the outputted object from scrapeVarCross() and automatically writes, returns, and exports (.txt) lavaan syntax for specifying dyadic configural, loading, intercept, and residual invariant hierarchical models.

Usage

```

scriptHier(
  dvn,
  scaleset = "FF",
  lvname = "X",
  lvar = "X",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = c("variances", "means"),
  constr_group_meas = NULL,
  constr_group_struct = NULL,
  writeTo = NULL,
  fileName = NULL,
  outputType = "lavaan script"
)

```

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 the latent variable in lavaan syntax
lvar	Input character to specify whether the latent variable represents "X" or "Y" in the model. Default is "X". This argument controls parameter labeling and which indicator variables are used from the dvn.
constr_dy_meas	Input character vector detailing which measurement model parameters to constrain across dyad members.
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 correlated dyadic factors model), but user can specify any combination of "variances" and "means", or "none".
constr_group_meas	Optional character vector detailing which measurement model parameters to constrain across groups. Valid values: "loadings", "intercepts", "residuals", or "none". Default NULL = single-group. Requires dvn from scrapeVarCross(..., group = "varname").
constr_group_struct	Optional character vector detailing which structural model parameters to constrain across groups. Valid values: "variances", "means", or "none". Default NULL = single-group. Requires dvn from scrapeVarCross(..., group = "varname").
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created

	by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
<code>fileName</code>	A character string specifying a desired base name for the output file. The default is <code>NULL</code> . The specified name will be automatically appended with the appropriate file extension (e.g., <code>.txt</code> for lavaan scripts, <code>.rtf</code> for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.
<code>outputType</code>	Character string specifying the type of output to return. Options are "lavaan script" (default) to return a character object of lavaan syntax that can be passed immediately to lavaan functions, or "syntax components" to return a structured list of model components.

Details

- By default, many `dySEM::` functions (including `scriptHier()`) 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 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 (when `outputType = "lavaan script"`), or a structured list of model components (when `outputType = "syntax components"`).

See Also

[scrapeVarCross](#) which this function relies on.

Other uni-construct script-writing functions: [scriptBifac\(\)](#), [scriptCor\(\)](#), [scriptUni\(\)](#)

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 <- scriptHier(  
  dvn,  
  scaleset = "FF",  
  lvname = "Sat"  
)  
  
sat.lvars.script <- scriptHier(  
  dvn,  
  scaleset = "FF",  
  lvname = "Sat",  
  constr_dy_meas = "loadings",  
  constr_dy_struct = "variances"  
)  
  
sat.resids.script <- scriptHier(  
  dvn,  
  scaleset = "FF",  
  lvname = "Sat",  
  constr_dy_meas = c("loadings", "intercepts", "residuals"),  
  constr_dy_struct = "none",  
  writeTo = tempdir(),  
  fileName = "dHier_residual"  
)  
  
sat.ints.script <- scriptHier(  
  dvn,  
  scaleset = "FF",  
  lvname = "Sat",  
  constr_dy_meas = c("loadings", "intercepts"),  
  constr_dy_struct = "none",  
  writeTo = tempdir(),  
  fileName = "dHier_intercept"  
)  
  
sat.loads.script <- scriptHier(  
  dvn,  
  scaleset = "FF",  
  lvname = "Sat",  
  constr_dy_meas = c("loadings"),  
  constr_dy_struct = "none",  
  writeTo = tempdir(),  
  fileName = "dHier_loading"  
)  
  
sat.config.script <- scriptHier(  
  dvn,  
  scaleset = "FF",  
  lvname = "Sat",  
  constr_dy_meas = "none",  
  constr_dy_struct = "none",  
  writeTo = tempdir(),  
  fileName = "dHier_configural"  
)
```

scriptINULL	<i>A Function That Writes, Saves, and Exports Syntax for Fitting the I-NULL model for indistinguishable dyads</i>
-------------	---

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",
  lvynname = NULL,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

<code>dvn</code>	input dvn list from <code>scrapeVarCross</code>
<code>lvxname</code>	input character to (arbitrarily) name X LV in lavaan syntax
<code>lvynname</code>	(optional) input character to (arbitrarily) name Y LV in lavaan syntax
<code>writeTo</code>	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
<code>fileName</code>	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). 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 indistinguishable script-writing functions: [scriptISAT\(\)](#)

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", lvynname = NULL, writeTo = NULL, fileName = NULL)
```

Arguments

dvn	input dvn list from <code>scrapeVarCross</code>
lvxname	input character to (arbitrarily) name X LV in lavaan syntax
lvynname	(optional) input character to (arbitrarily) name X LV in lavaan syntax
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). 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 indistinguishable script-writing functions: [scriptINULL\(\)](#)

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,
  lvynname,
  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 <code>scrapeVarCross</code>
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
lvynname	input character to (arbitrarily) name LV Y in lavaan syntax

<code>constr_dy_x_meas</code>	input character vector detailing which measurement model parameters to constrain across dyad members for latent X. Default is <code>c("loadings", "intercepts", "residuals")</code> , but user can specify any combination of "loadings", "intercepts", and "residuals", or "none" to specify an otherwise unconstrained dyadic configural invariance model
<code>constr_dy_x_struct</code>	input character vector detailing which structural model parameters to constrain across dyad members for latent X. Default is <code>c("variances", "means")</code> , but user can specify any combination of "variances" and "means", or "none".
<code>constr_dy_y_meas</code>	input character vector detailing which measurement model parameters to constrain across dyad members for latent Y. Default is <code>c("loadings", "intercepts", "residuals")</code> , but user can specify any combination of "loadings", "intercepts", and "residuals", or "none" to specify an otherwise unconstrained dyadic configural invariance model
<code>constr_dy_y_struct</code>	input character vector detailing which structural model parameters to constrain across dyad members for latent Y. Default is <code>c("variances", "means")</code> , but user can specify any combination of "variances" and "means", or "none".
<code>constr_dy_xy_struct</code>	input character vector detailing which structural model parameters to constrain for modeling the predictive association(s) between partners' latent x and y. Default is <code>c("actors", "partners")</code> , but users can also specify "all", "actors_zero", "partners_zero", or "none".
<code>model</code>	Deprecated input character used to specify which level of invariance is modeled. Users should rely upon <code>constr_dy_x_meas/constr_dy_y_meas</code> and <code>constr_dy_x_struct/constr_dy_y_struct</code> instead, for making constraints to the measurement and/or structural portions of the model for latent x and y.
<code>equate</code>	Deprecated input character to specify which type of structural parameters are constrained to equivalency between partners. Users should rely upon <code>constr_dy_xy_struct</code> for making constraints to the structural portion of the model for associative relationship between latent x and y.
<code>est_k</code>	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.
<code>writeTo</code>	A character string specifying a directory path to where the output file(s) should be saved. If set to <code> "." </code> , the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, <code>writeTo</code> is only relevant if <code>gtTab = TRUE</code> .
<code>fileName</code>	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., <code> .txt </code> for lavaan scripts, <code> .rtf </code> for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). 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 bi-construct script-writing functions: [scriptAPIM\(\)](#), [scriptBiDy\(\)](#), [scriptCFM\(\)](#), [scriptObsAPIM\(\)](#), [scriptTwoCross\(\)](#)

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

Observed Actor-Partner Interdependence Model (lavaan syntax)

Description

Writes lavaan syntax for an APIM with one observed (or composite) X and Y per partner. Structural constraints mirror [scriptAPIM\(\)](#) (dyadic variances/means, actor/partner paths, optional orthogonal dyadic covariances).

Usage

```
scriptObsAPIM(
  X1 = NULL,
  Y1 = NULL,
  X2 = NULL,
  Y2 = NULL,
  constr_dy_x_struct = c("variances", "means"),
  constr_dy_y_struct = c("variances", "means"),
  constr_dy_xy_struct = c("actors", "partners"),
  includeMeanStruct = FALSE,
  equate = lifecycle::deprecated(),
  k = lifecycle::deprecated(),
  est_k = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

X1, X2, Y1, Y2	Character names of variables for partners 1 and 2.
constr_dy_x_struct	Character vector: any of "variances", "means", "orthogonal", and/or "none". Defaults match scriptAPIM() . "means" is applied only when includeMeanStruct is TRUE (same behavior as scriptAPIM()). "orthogonal" fixes the dyadic covariance between X1 and X2 to zero.
constr_dy_y_struct	Same allowed values as constr_dy_x_struct, for the Y side. "variances" equates residual variances of Y1 and Y2; "orthogonal" fixes the residual covariance between Y1 and Y2 to zero.
constr_dy_xy_struct	Character vector: "actors", "partners", "all", "actors_zero", "partners_zero", and/or "none" (same semantics as scriptAPIM()).
includeMeanStruct	Logical; if TRUE, mean structure is included and "means" in constr_dy_x_struct / constr_dy_y_struct equates exogenous means and regression intercepts, respectively.
equate	[Deprecated] Use constr_dy_xy_struct instead. If supplied, maps none → "none", actor → c("actors"), partner → c("partners"), all → c("actors", "partners").
k	[Deprecated] Use est_k instead.
est_k	Logical; if TRUE, Kenny & Ledermann's (2010) k ratio(s) are defined in the syntax (see scriptAPIM()).
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by <code>tempdir()</code> . When dealing with tabular output, writeTo is only relevant if <code>gtTab = TRUE</code> .
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when <code>gtTab = TRUE</code> , or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.

Value

Character string of lavaan model syntax.

See Also

[scriptAPIM\(\)](#)

Other bi-construct script-writing functions: [scriptAPIM\(\)](#), [scriptBiDy\(\)](#), [scriptCFM\(\)](#), [scriptMIM\(\)](#), [scriptTwoCross\(\)](#)

Examples

```
scriptObsAPIM(
  X1 = "SexSatA", Y1 = "RelSatA",
  X2 = "SexSatB", Y2 = "RelSatB",
  constr_dy_xy_struct = "none",
  constr_dy_x_struct = "none",
  constr_dy_y_struct = "none",
  writeTo = tempdir(),
  fileName = "obsAPIM_script"
)
```

scriptPopAPIM

Population lavaan syntax for a latent indistinguishable APIM

Description

Writes a fully specified population model with fixed numeric parameters for use with `lavaan::simulateData()`. Measurement blocks follow the same conventions as `scriptPopCor()` for each construct; structural paths use a single actor and partner coefficient on both partners' equations.

Usage

```
scriptPopAPIM(dvn, lvxname, lvyname, pop_values)
```

Arguments

dvn	List from <code>build_dvn()</code> or <code>scrapeVarCross()</code> with X and Y indicators (p1yvarnames, p2yvarnames, yindper required).
lvxname, lvyname	Base names for latent X and Y (partner suffixes from dvn\$dist1 / dvn\$dist2 are appended, as in <code>scriptAPIM()</code>).
pop_values	Named list; see Population values below.

Value

Single string of lavaan model syntax.

Population values

X measurement (lengths nx = dvn\$xindep): x_loadings_p1, x_loadings_p2 (optional recycle), x_residuals_p1, x_residuals_p2, x_coresids, optional x_intercepts_p1 / x_intercepts_p2.

Y measurement (ny = dvn\$yindper): y_loadings_p1, y_loadings_p2, y_residuals_p1, y_residuals_p2, y_coresids, optional y_intercepts_.*.

Latent X: x_lv_var (scalar or length 2, default c(1, 1)), x_lv_cov (default 0), x_lv_means (optional, default c(0, 0)).

Structural: actor, partner (numeric), y_lv_var (scalar or length 2, default c(0.5, 0.5)), y_lv_cov (default 0; covariance between latent Y residuals).

Vectors of length 1 are recycled to nx or ny for measurement slots.

See Also

Other population-scripting functions: [scriptPopCor\(\)](#), [scriptPopUni\(\)](#)

scriptPopCor

Write Population lavaan Syntax for Correlated Dyadic CFA

Description

Produces lavaan syntax with fixed numeric values for data generation in Monte Carlo simulations. Used with `lavaan::simulateData()`.

Usage

```
scriptPopCor(dvn, lvname = "X", pop_values, lvar = "X")
```

Arguments

dvn	Input dvn list from <code>build_dvn()</code> or <code>scrapeVarCross()</code>
lvname	Character name for the latent variable
pop_values	Named list of population parameter values: <ul style="list-style-type: none"> • <code>loadings_p1</code>: numeric vector for partner 1 (length = <code>n_items</code>) • <code>loadings_p2</code>: numeric vector for partner 2 (length = <code>n_items</code>). If <code>NULL</code>, uses <code>loadings_p1</code>. • <code>residuals_p1</code>: numeric vector (length = <code>n_items</code>) • <code>residuals_p2</code>: numeric vector (length = <code>n_items</code>). If <code>NULL</code>, uses <code>residuals_p1</code>. • <code>coresids</code>: numeric vector (length = <code>n_items</code>) • <code>intercepts_p1</code>: optional (default 0) • <code>intercepts_p2</code>: optional (default 0) • <code>lv_var</code>: latent variances; scalar (both=1) or length-2 vector • <code>lv_cov</code>: latent covariance • <code>lv_means</code>: latent means; scalar (both=0) or length-2 vector
lvar	Character "X" or "Y". Default "X".

Value

Character string of lavaan syntax

See Also

Other population-scripting functions: [scriptPopAPIM\(\)](#), [scriptPopUni\(\)](#)

Examples

```
dvn <- dySEM::build_dvn(n_items_x = 3, x_stem = "x", x_delim1 = "", x_delim2 = "_",
  distinguish_1 = "A", distinguish_2 = "B")
scriptPopCor(dvn, "X", list(
  loadings_p1 = c(.75, .80, .55),
  loadings_p2 = c(.70, .85, .50),
  residuals_p1 = c(.44, .36, .70),
  residuals_p2 = c(.51, .28, .75),
  coresids = c(.10, .12, .08),
  lv_cov = 0.3
))
```

scriptPopUni

*Write Population lavaan Syntax for Unidimensional Dyadic CFA***Description**

Produces lavaan syntax with fixed numeric values for data generation in Monte Carlo simulations. Used with `lavaan::simulateData()`.

Usage

```
scriptPopUni(dvn, lvname = "X", pop_values, lvar = "X")
```

Arguments

dvn	Input dvn list from <code>build_dvn()</code> or <code>scrapeVarCross()</code>
lvname	Character name for the latent variable
pop_values	Named list of population parameter values: <ul style="list-style-type: none"> • <code>loadings</code>: numeric vector of loadings (<code>length = n_items</code>) • <code>residuals</code>: numeric vector of residual variances (<code>length = n_items</code>) • <code>coresids</code>: numeric vector of residual covariances (<code>length = n_items</code>) • <code>intercepts</code>: optional; numeric vector (default 0 for all) • <code>lv_var</code>: latent variance (default 1) • <code>lv_mean</code>: latent mean (default 0)
lvar	Character "X" or "Y". Default "X".

Value

Character string of lavaan syntax

See Also

Other population-scripting functions: [scriptPopAPIM\(\)](#), [scriptPopCor\(\)](#)

Examples

```
dvn <- dySEM::build_dvn(n_items_x = 3, x_stem = "x", x_delim1 = "", x_delim2 = "_",
  distinguish_1 = "A", distinguish_2 = "B")
scriptPopUni(dvn, "X", list(
  loadings = c(.75, .80, .55),
  residuals = c(.44, .36, .70),
  coresids = c(.10, .12, .08)
))
```

scriptTwoCross	<i>A Function That Writes, Saves, and Exports Syntax for Fitting Boutique Two-Construct Dyadic SEM Models</i>
----------------	---

Description

This function takes the outputted object from `scrapeVarCross()` with both X and Y indicators and automatically writes, returns, and exports (.txt) lavaan syntax for specifying a two-construct model where the measurement structure of X and Y can differ (e.g., X as correlated factors, Y as unidimensional).

Usage

```
scriptTwoCross(
  dvn,
  x_scaleset = "FF",
  y_scaleset = "FF",
  lvxname = "X",
  lvyname = "Y",
  x_model = "cor",
  y_model = "cor",
  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 = "free",
  includeMeanStruct = FALSE,
  writeTo = NULL,
  fileName = NULL
)
```

Arguments

dvn	Input dvn list from <code>scrapeVarCross()</code> containing information for both X and Y (i.e., must include both <code>x_stem</code> and <code>y_stem</code> arguments when calling <code>scrapeVarCross</code>).
x_scaleset	Character string specifying how to set the scale of latent X. Options are "FF" (fixed-factor) or "MV" (marker variable). Default is "FF".

y_scaleset	Character string specifying how to set the scale of latent Y. Options are "FF" (fixed-factor) or "MV" (marker variable). Default is "FF".
lvxname	Input character to (arbitrarily) name latent variable X in lavaan syntax.
lvyname	Input character to (arbitrarily) name latent variable Y in lavaan syntax.
x_model	Character string specifying the measurement model for X. Options are "uni" (unidimensional), "cor" (correlated factors), "hier" (hierarchical), or "bifactor".
y_model	Character string specifying the measurement model for Y. Same options as x_model.
constr_dy_x_meas	Input character vector detailing which measurement model parameters to constrain across dyad members for latent X.
constr_dy_x_struct	Input character vector detailing which structural model parameters to constrain across dyad members for latent X.
constr_dy_y_meas	Input character vector detailing which measurement model parameters to constrain across dyad members for latent Y.
constr_dy_y_struct	Input character vector detailing which structural model parameters to constrain across dyad members for latent Y.
constr_dy_xy_struct	Character string for the X-to-Y regression paths. Options are "free" (default; all paths freely estimated) or "zero" (all paths fixed to zero).
includeMeanStruct	Logical for whether to include the mean structure (intercepts and latent means) in the model. Defaults to FALSE.
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). 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; [scriptUni](#), [scriptCor](#), [scriptHier](#), [scriptBifac](#) for the underlying scripts; [scriptAPIM](#) for a related two-construct function.

Other bi-construct script-writing functions: [scriptAPIM\(\)](#), [scriptBiDy\(\)](#), [scriptCFM\(\)](#), [scriptMIM\(\)](#), [scriptObsAPIM\(\)](#)

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 = "_"
)

# Correlated X, unidimensional Y
scriptTwoCross(dvn, lvxname = "Sat", lvynname = "Com",
  x_model = "cor", y_model = "uni")

# Both correlated
scriptTwoCross(dvn, lvxname = "Sat", lvynname = "Com",
  x_model = "cor", y_model = "cor",
  writeTo = tempdir(), fileName = "twoCross_cor_cor")
```

scriptUni

A Function That Writes, Saves, and Exports Syntax for Fitting Unidimensional Dyadic Factor Models

Description

This function takes the outputted object from `scrapeVarCross()` and automatically writes, returns, and exports (.txt) lavaan syntax for specifying dyadic configural, loading, intercept, and residual invariant one-factor models.

Usage

```
scriptUni(
  dvn,
  scaleset = "FF",
  lvname = "X",
  lvar = "X",
  constr_dy_meas = c("loadings", "intercepts", "residuals"),
  constr_dy_struct = "none",
  constr_group_meas = NULL,
  constr_group_struct = NULL,
  writeTo = NULL,
  fileName = NULL,
  outputType = "lavaan script"
)
```

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 the latent variable in lavaan syntax.
lvar	Input character to specify whether the latent variable represents "X" or "Y" in the model. Default is "X". This argument controls parameter labeling (e.g., lx vs ly for loadings, tx vs ty for intercepts).
constr_dy_meas	Input character vector detailing which measurement model parameters to constrain across dyad members.
constr_dy_struct	Input character vector detailing which structural model parameters to constrain across dyad members. Note: Within the context of scriptUni(), constr_dy_struct is irrelevant , as the unidimensional dyadic factor model assumes a single latent variable shared by both partners, leaving no structural parameters to constrain across the modeled dyad members. For consistency with other scripter functions, constr_dy_struct is included as an argument, but defaults to "none".
constr_group_meas	Optional character vector for cross-group measurement constraints (e.g. "loadings", "intercepts", "residuals", "none"). When NULL (default), single-group model. When non-NULL, requires dvn from scrapeVarCross(..., group = "varname").
constr_group_struct	Optional character vector for cross-group structural constraints (e.g. "variances", "means", "none"). When NULL (default), single-group model.
writeTo	A character string specifying a directory path to where the output file(s) should be saved. If set to ".", the file(s) will be written to the current working directory. The default is NULL, and examples use a temporary directory created by tempdir(). When dealing with tabular output, writeTo is only relevant if gtTab = TRUE.
fileName	A character string specifying a desired base name for the output file. The default is NULL. The specified name will be automatically appended with the appropriate file extension (e.g., .txt for lavaan scripts, .rtf for tabular output when gtTab = TRUE, or other extensions as appropriate for the output type). If a file with the same name already exists in the user's chosen directory, it will be overwritten.
outputType	Character string specifying the type of output to return. Options are "lavaan script" (default) to return a character object of lavaan syntax that can be passed immediately to lavaan functions, or "syntax components" to return a structured list of model components.

Details

- Users do not need to modify constr_dy_struct when using scriptUni().
- By default, many dySEM: : functions (including scriptUni()) 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 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 (when `outputType = "lavaan script"`), or a structured list of model components (when `outputType = "syntax components"`).

See Also

[scrapeVarCross](#) which this function relies on.

Other uni-construct script-writing functions: [scriptBifac\(\)](#), [scriptCor\(\)](#), [scriptHier\(\)](#)

Examples

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

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

sat.ints.script <- scriptUni(
  dvn,
  scaleset = "FF",
  lvname = "Sat",
  constr_dy_meas = c("loadings", "intercepts"),
  constr_dy_struct = "none",
  writeTo = tempdir(),
  fileName = "dUni_intercept"
)

sat.loads.script <- scriptUni(
```

```
    dvn,  
    scaleset = "FF",  
    lvname = "Sat",  
    constr_dy_meas = c("loadings"),  
    constr_dy_struct = "none",  
    writeTo = tempdir(),  
    fileName = "dUni_loading"  
  )  
  
sat.config.script <- scriptUni(  
  dvn,  
  scaleset = "FF",  
  lvname = "Sat",  
  constr_dy_meas = "none",  
  constr_dy_struct = "none",  
  writeTo = tempdir(),  
  fileName = "dUni_configural"  
)
```

Index

- * **bi-construct script-writing functions**
 - scriptAPIM, 35
 - scriptBiDy, 37
 - scriptCFM, 47
 - scriptMIM, 59
 - scriptObsAPIM, 61
 - scriptTwoCross, 66
 - * **datasets**
 - commitmentM, 4
 - commitmentQ, 5
 - DRES, 7
 - imsM, 13
 - pnrqM, 30
 - prqcQ, 31
 - * **indistinguishable script-writing functions**
 - scriptINULL, 57
 - scriptISAT, 58
 - * **multi-construct script-writing functions**
 - scriptCFA, 43
 - scriptDyEFA, 52
 - * **output functions**
 - outputInvarItem, 20
 - * **population-scripting functions**
 - scriptPopAPIM, 63
 - scriptPopCor, 64
 - scriptPopUni, 65
 - * **supplemental model calculators**
 - getDydmacs, 9
 - getDyReliability, 10
 - getIndistFit, 10
 - * **uni-construct script-writing functions**
 - scriptBifac, 40
 - scriptCor, 49
 - scriptHier, 53
 - scriptUni, 68
 - * **variable-scraping functions**
 - build_dvn, 3
 - scrapeVarCross, 32
- build_dvn, 3, 34
- build_dvn(), 63
- commitmentM, 4
- commitmentQ, 5
- DRES, 7
- getConstraintTests, 8
- getDydmacs, 9, 10, 11
- getDyReliability, 9, 10, 11
- getIndistFit, 9, 10, 10
- getInvarCompTable, 12
- imsM, 13
- launchScripter, 15
- lavaan::simulateData(), 63
- outputConstraintTab, 16
- outputInvarCompTab, 17
- outputInvarItem, 19
- outputModel, 21
- outputParamFig, 23
- outputParamTab, 25
- outputUniConstructComp, 27
- pnrqM, 30
- prqcQ, 31
- runApp, 15
- scrapeVarCross, 4, 32, 37, 41, 45, 49, 51, 53, 55, 57, 58, 61, 67, 70
- scrapeVarCross(), 63
- scriptAPIM, 35, 39, 49, 61, 62, 67
- scriptAPIM(), 61–63
- scriptBiDy, 37, 37, 49, 61, 62, 67
- scriptBifac, 40, 51, 55, 67, 70
- scriptCFA, 43, 53
- scriptCFM, 37, 39, 47, 61, 62, 67
- scriptCor, 41, 49, 55, 67, 70

scriptDyEFA, [45](#), [52](#)
scriptHier, [41](#), [51](#), [53](#), [67](#), [70](#)
scriptINULL, [57](#), [58](#)
scriptISAT, [57](#), [58](#)
scriptMIM, [37](#), [39](#), [49](#), [59](#), [62](#), [67](#)
scriptMultiCor (scriptCFA), [43](#)
scriptObsAPIM, [37](#), [39](#), [49](#), [61](#), [61](#), [67](#)
scriptPopAPIM, [63](#), [64](#), [65](#)
scriptPopCor, [64](#), [64](#), [65](#)
scriptPopCor(), [63](#)
scriptPopUni, [64](#), [65](#)
scriptTwoCross, [37](#), [39](#), [49](#), [61](#), [62](#), [66](#)
scriptUni, [41](#), [51](#), [55](#), [67](#), [68](#)