Package: psycModel (via r-universe)

November 4, 2024

Type Package

Title Integrated Toolkit for Psychological Analysis and Modeling in R

Version 0.5.0.9000

Description A beginner-friendly R package for modeling in psychology or related field. It allows fitting models, plotting, checking goodness of fit, and model assumption violations all in one place. It also produces beautiful and easy-to-read output.

License GPL (>= 3)

URL https://github.com/jasonmoy28/psycModel

Depends R (>= 3.2)

Imports dplyr, ggplot2, glue, insight, lavaan, lifecycle, lme4, lmerTest, parameters, patchwork, performance, psych, rlang (>= 0.1.2), stringr, tibble, tidyr, utils, tidyselect, effects, MASS,

Suggests correlation, covr, cowplot, fansi, ggrepel, GPArotation, gridExtra, interactions, knitr, nFactors, nlme, pagedown, qqplotr, rmarkdown, roxygen2, sandwich, see, semPlot, spelling, testthat (>= 3.0.0), lavaSearch2, scales

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Language en-US

Repository https://jasonmoy28.r-universe.dev

RemoteUrl https://github.com/jasonmoy28/psycmodel

RemoteRef HEAD

RemoteSha c9800ea1930987746dbc84ea560ad5355e217988

Contents

acitelli . . . 2
anova_plot . . . 3
APIM_sem . . . 4
APIM_table . . . 6
cfa_groupwise . . . 7
cfa_summary . . . 8
compare_fit . . . 10
cor_test . . . 11
cronbach_alpha . . . 13
descriptive_table . . . 14
efa_summary . . . 15
get_interaction_term . . . 16
get_predict_df . . . 17
glm_model . . . 17
glm_model_explore . . . 18
html_to_pdf . . . 20
interaction_plot . . . 21
knit_to_Rmd . . . 23
label_name . . . 24
lme_model_explore . . . 25
lme_model_table . . . 26
lm_model . . . 28
lm_model_explore . . . 29
lm_model_summary . . . 31
lm_model_table . . . 33
measurement_invariance . . . 35
mediation . . . 38
mediation_summary . . . 39
model_summary . . . 41
polynomial_regression_plot . . . 42
popular . . . 43
reliability_summary . . . 44
simple_slope . . . 45
three_way_interaction_plot . . . 46
two_way_interaction_plot . . . 48

Index **50**

`acitelli` *Acitelli*

Description

Acitelli, L. K. (1997). Sampling couples to understand them: Mixing the theoretical with the practical. Journal of Social and Personal Relationships, 14, 243-261.

Usage

```
acitelli
```

Format

A data frame with 296 rows and 16 variables:

CoupleID each couple has 1 couple id

Yearsmar each couple has 1 value for years married (centered; 11.214 subtracted)

Partnum A variable to distinguish the two members (1 and 2)

Gender_A -1 Female; 1 Male

SelfPos_A Mean of positive self variables (uncentered)

OtherPos_A Mean of positive partner variables (uncentered)(both variables the average of Cooperative, Mature, Friendly, Hard Working and Care about Others)

Satisfaction_A Mean of four variables (theoretical range from 1 to 4)

Tension_A Mean of two disagreement and tension (theoretical range from 1 to 4)

SimHob_A Perception of having similar hobbies (1 = very true; 0 = somewhat true; -1 = not at all true)

Gender_P -1 Female; 1 Male

SelfPos_P Mean of positive self variables (uncentered)

OtherPos_P Mean of positive partner variables (uncentered) (both variables the average of Cooperative, Mature, Friendly, Hard Working and Care about Others)

Satisfaction_P Mean of four variables (theoretical range from 1 to 4)

Tension_P Mean of two disagreement and tension (theoretical range from 1 to 4)

SimHob_P Perception of having similar hobbies (1 = very true; 0 = somewhat true; -1 = not at all true)

Gender String variable with Husband and Wife

Source

https://journals.sagepub.com/doi/10.1177/0265407597142006

---

anova_plot *ANOVA Plot*

Description

[Experimental]
Plot categorical variable with barplot. Continuous moderator are plotted at ± 1 SD from the mean.

Usage

```
anova_plot(model, predictor = NULL, graph_label_name = NULL, y_lim = NULL)
```

Arguments

model	fitted model (usually lm or aov object). Variables must be converted to correct data type before fitting the model. Specifically, continuous variables must be converted to type numeric and categorical variables to type factor.
predictor	predictor variable. Must specified for non-interaction plot and must not specify for interaction plot.
graph_label_name	vector or function. Vector should be passed in the form of c(response_var, predict_var1, predict_var2, ...). Function should be passed as a switch function that return the label based on the name passed (e.g., a switch function)
y_lim	the plot's upper and lower limit for the y-axis. Length of 2. Example: c(lower_limit, upper_limit)

Value

a ggplot object

Examples

```
# Main effect plot with 1 categorical variable
fit_1 = lavaan::HolzingerSwineford1939 %>%
  dplyr::mutate(school = as.factor(school)) %>%
  lm(data = ., grade ~ school)
anova_plot(fit_1,predictor = school)

# Interaction effect plot with 2 categorical variables
fit_2 = lavaan::HolzingerSwineford1939 %>%
  dplyr::mutate(dplyr::across(c(school,sex),as.factor)) %>%
  lm(data = ., grade ~ school*sex)
anova_plot(fit_2)

# Interaction effect plot with 1 categorical variable and 1 continuous variable
fit_3 = lavaan::HolzingerSwineford1939 %>%
  dplyr::mutate(school = as.factor(school)) %>%
  dplyr::mutate(ageyr = as.numeric(ageyr)) %>%
  lm(data = ., grade ~ ageyr*school)
anova_plot(fit_3)
```

APIM_sem *APIM Actor-Partner Interdependence Model (SEM)*

Description

[Stable]

Usage

```
APIM_sem(
  data,
  mod_type,
  predictor_a,
  predictor_p,
  outcome_a,
  outcome_p,
  med_a = NULL,
  med_p = NULL,
  mod_a = NULL,
  mod_p = NULL,
  bootstrap = NULL,
  standardized = FALSE,
  return_result = FALSE,
  quite = FALSE
)
```

Arguments

`data`	data frame object
`mod_type`	options are "simple" (main effect), "med" (mediation), and "mod" (moderation)
`predictor_a`	predictor variable name for actor
`predictor_p`	predictor variable name for partner
`outcome_a`	dependent variable name for actor
`outcome_p`	dependent variable name for partner
`med_a`	mediation variable name for actor
`med_p`	mediation variable name for partner
`mod_a`	moderation variable name for actor
`mod_p`	moderation variable name for partner
`bootstrap`	number of bootstrapping (e.g., 5000). Default is not using bootstrap
`standardized`	standardized coefficient
`return_result`	return `lavaan::parameterestimates()`. Default is `FALSE`
`quite`	suppress printing output. Default is `FALSE`

Details

Actor-partner interdependence model using SEM approach (with lavaan). Indistinguishable dyads only. Results should be the same as those from Kenny (2015a, 2015b).

Value

data.frame from `lavaan::parameterestimates()`

References

Kenny, D. A. (2015, October). An interactive tool for the estimation and testing mediation in the Actor-Partner Interdependence Model using structural equation modeling. Computer software. Available from https://davidakenny.shinyapps.io/APIMeM/. Kenny, D. A. (2015, October). An interactive tool for the estimation and testing moderation in the Actor-Partner Interdependence Model using structural equation modeling. Computer software. Available from https://davidakenny.shinyapps.io/APIMoM/. Stas, L, Kenny, D. A., Mayer, A., & Loeys, T. (2018). Giving Dyadic Data Analysis Away: A User-Friendly App for Actor-Partner Interdependence Models. Personal Relationships, 25 (1), 103-119. https://doi.org/10.1111/pere.12230.

Examples

```
APIM_sem(data = acitelli,
        predictor_a = 'Tension_A',
        predictor_p = 'Tension_P',
        outcome_a = 'Satisfaction_A',
        outcome_p = 'Satisfaction_P',
        mod_type = 'simple')
```

APIM_table *APIM Actor-Partner Interdependence Model Table with Multiple Moderators*

Description

[Stable]
Actor-partner interdependence model that test multiple moderators simultaneously.

Usage

```
APIM_table(
  data,
  predictor_a,
  predictor_p,
  outcome_a,
  outcome_p,
  mod_a = NULL,
  mod_p = NULL,
  mod_type = "mod",
  return_result = FALSE
)
```

Arguments

data	data frame object
predictor_a	predictor variable name for actor

predictor_p	predictor variable name for partner
outcome_a	dependent variable name for actor
outcome_p	dependent variable name for partner
mod_a	moderation variable name for actor. Support dplyr::select() syntax.
mod_p	moderation variable name for partner. Support dplyr::select() syntax.
mod_type	only 'mod' is supported for now
return_result	return lavaan::parameterestimates(). Default is FALSE

Value

data.frame of the APIM table

Examples

```
APIM_table(data = acitelli,
           predictor_a = 'Tension_A',
           predictor_p = 'Tension_P',
           outcome_a = 'Satisfaction_A',
           outcome_p = 'Satisfaction_P',
           mod_a = c('SelfPos_A','OtherPos_A','SimHob_A'),
           mod_p = c('SelfPos_P','OtherPos_P','SimHob_P'))
```

---

cfa_groupwise *Confirmatory Factor Analysis (groupwise)*

Description

[Superseded]
This function will run N number of CFA where N = length(group), and report the fit measures of CFA in each group. The function is intended to help you get a better understanding of which group has abnormal fit indicator

Usage

```
cfa_groupwise(data, ..., group, model = NULL, ordered = FALSE)
```

Arguments

data	data frame
...	CFA items. Support dplyr::select() syntax.
group	character. group variable. Support dplyr::select() syntax.
model	explicit lavaan model. Must be specify with model = lavaan_model_syntax. **[Experimental]**
ordered	logical. default is FALSE. If it is set to TRUE, lavaan will treat it as a ordinal variable and use DWLS instead of ML

Details

All argument must be explicitly specified. If not, all arguments will be treated as CFA items

Value

a `data.frame` with group-wise CFA result

Examples

```
# The example is used as the illustration of the function output only.
# It does not imply the data is appropriate for the analysis.
cfa_groupwise(
  data = lavaan::HolzingerSwineford1939,
  group = "school",
  x1:x3,
  x4:x6,
  x7:x9
)
```

`cfa_summary` *Confirmatory Factor Analysis*

Description

[Stable]
The function fits a CFA model using the `lavaan::cfa()`. Users can fit single and multiple factors CFA, and it also supports multilevel CFA (by specifying the group). Users can fit the model by passing the items using `dplyr::select()` syntax or an explicit lavaan model for more versatile usage. All arguments (except the CFA items) must be explicitly named (e.g., model = your-model; see example for inappropriate behavior).

Usage

```
cfa_summary(
  data,
  ...,
  model = NULL,
  group = NULL,
  ordered = FALSE,
  digits = 3,
  estimator = "ML",
  model_covariance = TRUE,
  model_variance = TRUE,
  plot = TRUE,
  group_partial = NULL,
  streamline = FALSE,
  quite = FALSE,
  return_result = FALSE
)
```

Arguments

data	data frame
...	CFA items. Multi-factor CFA items should be separated by comma (as different argument). See below for examples. Support dplyr::select() syntax.
model	explicit lavaan model. Must be specify with model = lavaan_model_syntax. **[Experimental]**
group	optional character. used for multi-level CFA. the nested variable for multilevel dataset (e.g., Country). Support dplyr::select() syntax.
ordered	Default is FALSE. If it is set to TRUE, lavaan will treat it as a ordinal variable and use DWLS instead of ML
digits	number of digits to round to
estimator	estimator for lavaan. Default is ML
model_covariance	print model covariance. Default is TRUE
model_variance	print model variance. Default is TRUE
plot	print a path diagram. Default is TRUE
group_partial	Items for partial equivalence. The form should be c('DV =~ item1', 'DV =~ item2').
streamline	print streamlined output
quite	suppress printing output
return_result	If it is set to TRUE, it will return the lavaan model

Details

First, just like researchers have argued against p value of 0.05 is not a good cut-of, researchers have also argue against that fit indicies (more importantly, the cut-off criteria) are not completely representative of the goodness of fit. Nonetheless, you are required to report them if you are publishing an article anyway. I will summarize the general recommended cut-off criteria for CFA model below. Researchers consider models with CFI (Bentler, 1990) that is > 0.95 to be excellent fit (Hu & Bentler, 1999), and > 0.9 to be acceptable fit. Researchers considered a model is excellent fit if CFI > 0.95 (Hu & Bentler, 1999), RMSEA < 0.06 (Hu & Bentler, 1999), TLI > 0.95, SRMR < 0.08. The model is considered an acceptable fit if CFI > 0.9 and RMSEA < 0.08. I need some time to find all the relevant references, but this should be the general consensus.

Value

a lavaan object if return_result is TRUE

References

Hu, L., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. Structural Equation Modeling, 6, 1–55. https://doi.org/10.1080/10705519909540118

Examples

```
# REMEMBER, YOU MUST NAMED ALL ARGUMENT EXCEPT THE CFA ITEMS ARGUMENT
# Fitting a multilevel single factor CFA model
fit <- cfa_summary(
  data = lavaan::HolzingerSwineford1939,
  x1:x3,
  x4:x6,
  x7:x9,
  group = "sex",
  model_variance = FALSE, # do not print the model_variance
  model_covariance = FALSE # do not print the model_covariance
)


# Fitting a CFA model by passing explicit lavaan model (equivalent to the above model)
# Note in the below function how I added `model = ` in front of the lavaan model.
# Similarly, the same rule apply for all arguments (e.g., `ordered = FALSE` instead of just `FALSE`)

fit <- cfa_summary(
  model = "visual  =~ x1 + x2 + x3",
  data = lavaan::HolzingerSwineford1939,
  quite = TRUE # silence all output
)

## Not run:
# This will fail because I did not add `model = ` in front of the lavaan model.
# Therefore,you must add the tag in front of all arguments
# For example, `return_result = 'model'` instaed of `model`
cfa_summary("visual  =~ x1 + x2 + x3
             textual =~ x4 + x5 + x6
             speed   =~ x7 + x8 + x9 ",
  data = lavaan::HolzingerSwineford1939
)

## End(Not run)
```

---

compare_fit *Comparison of Model Fit*

Description

[Stable]
Compare the fit indices of models (see below for model support)

Usage

```
compare_fit(
  ...,
  digits = 3,
```

```
  quite = FALSE,
  streamline = FALSE,
  return_result = FALSE
)
```

Arguments

...	model. If it is a lavaan object, it will try to compute the measurement invariance. Other model types will be passed to performance::compare_performance().
digits	number of digits to round to
quite	suppress printing output
streamline	print streamlined output
return_result	If it is set to TRUE, it will return the the compare fit data frame.

Value

a dataframe with fit indices and change in fit indices

Examples

```
# lme model

fit1 <- lm_model(
  data = popular,
  response_variable = popular,
  predictor_var = c(sex, extrav)
)

fit2 <- lm_model(
  data = popular,
  response_variable = popular,
  predictor_var = c(sex, extrav),
  two_way_interaction_factor = c(sex, extrav)
)

compare_fit(fit1, fit2)

# see ?measurement_invariance for measurement invariance example
```

---

cor_test *Correlation table*

Description

[Stable]
This function uses the correlation::correlation() to generate the correlation table.

Usage

```
cor_test(
  data,
  cols,
  ...,
  digits = 3,
  show_p = FALSE,
  method = "pearson",
  p_adjust = "none",
  streamline = FALSE,
  verbose = TRUE,
  return_result = FALSE
)
```

Arguments

data	data frame
cols	correlation items. Support dplyr::select() syntax.
...	additional arguments passed to correlation::correlation(). See ?correlation::correlation. Note that the return data.frame from correlation::correlation() must contains r and p (e.g., passing baysesian = TRUE will not work)
digits	number of digits to round to
show_p	Default is FALSE. If TRUE, show the p-value in parenthesis.
method	Default is "pearson". Options are "kendall", "spearman","biserial", "polychoric", "tetrachoric", "biweight", "distance", "percentage", "blomqvist", "hoeffding", "gamma", "gaussian","shepherd", or "auto". See ?correlation::correlation for detail
p_adjust	Default is "holm". Options are "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "somers" or "none". See ?stats::p.adjust for more detail
streamline	print streamlined output.
verbose	deafult is TRUE.
return_result	If it is set to TRUE, it will return the data frame of the correlation table

Value

a data.frame of the correlation table

Examples

```
cor_test(iris, where(is.numeric))
```

`cronbach_alpha` *Cronbach alpha*

Description

[Stable]
Computing the Cronbach alphas for multiple factors.

Usage

```
cronbach_alpha(
  ...,
  data,
  var_name,
  group = NULL,
  quite = FALSE,
  return_result = FALSE
)
```

Arguments

`...`	Items. Group each latent factors using c() with when computing Cronbach alpha for 2+ factors (see example below)
`data`	`data.frame`. Must specify
`var_name`	character or a vector of characters. The order of `var_name` must be same as the order of the `...`
`group`	optional character. Specify this argument for computing Cronbach alpha for group separetely
`quite`	suppress printing output
`return_result`	If it is set to `TRUE`, it will return a `dataframe` object

Value

a `data.frame` object if return_result is `TRUE`

Examples

```
cronbach_alpha(
  data = lavaan::HolzingerSwineford1939,
  var_name = c('Visual','Textual','Speed'),
  c(x1,x2,x3), # one way to pass the items of a factor is by wrapping it with c()
  x4:x6, # another way to pass the items is use tidyselect syntax
  x7:x9)
```

`descriptive_table` *Descriptive Statistics Table*

Description

[Stable]
This function generates a table of descriptive statistics (mainly using `psych::describe()`) and or a correlation table. User can export this to a csv file (optionally, using the file_path argument). Users can open the csv file with MS Excel then copy and paste the table into MS Word table.

Usage

```
descriptive_table(
  data,
  cols,
  ...,
  digits = 3,
  descriptive_indicator = c("mean", "sd", "cor"),
  file_path = NULL,
  streamline = FALSE,
  quite = FALSE,
  show_p = FALSE,
  return_result = FALSE
)
```

Arguments

`data`	`data.frame`
`cols`	column(s) need to be included in the table. Support `dplyr::select()` syntax.
`...`	additional arguments passed to cor_test. See ?cor_test.
`digits`	number of digit for the descriptive table
`descriptive_indicator`	Default is mean, sd, cor. Options are missing (missing value count), non_missing (non-missing value count), cor (correlation table), n, mean, sd, median, trimmed (trimmed mean), median, mad (median absolute deviation from the median), min, max, range, skew, kurtosis, se (standard error)
`file_path`	file path for export. The function will implicitly pass this argument to the write.csv(file = file_path)
`streamline`	print streamlined output
`quite`	suppress printing output
`show_p`	Default is `FALSE`. If `TRUE`, show the p-value in parenthesis for correlations.
`return_result`	If it is set to `TRUE`, it will return the data frame of the descriptive table

Value

a data.frame of the descriptive table

Examples

```
descriptive_table(iris, cols = where(is.numeric)) # all numeric columns

descriptive_table(iris,
  cols = where(is.numeric),
  # get missing count, non-missing count, and mean & sd & correlation table
  descriptive_indicator = c("missing", "non_missing", "mean", "sd", "cor")
)
```

---

efa_summary *Exploratory Factor Analysis*

Description

[Stable]
The function is used to fit a exploratory factor analysis model. It will first find the optimal number of factors using parameters::n_factors. Once the optimal number of factor is determined, the function will fit the model using psych::fa(). Optionally, you can request a post-hoc CFA model based on the EFA model which gives you more fit indexes (e.g., CFI, RMSEA, TLI)

Usage

```
efa_summary(
  data,
  cols,
  rotation = "varimax",
  optimal_factor_method = FALSE,
  efa_plot = TRUE,
  digits = 3,
  n_factor = NULL,
  post_hoc_cfa = FALSE,
  quite = FALSE,
  streamline = FALSE,
  return_result = FALSE
)
```

Arguments

data	data.frame
cols	columns. Support dplyr::select() syntax.
rotation	the rotation to use in estimation. Default is 'oblimin'. Options are 'none', 'varimax', 'quartimax', 'promax', 'oblimin', or 'simplimax'

optimal_factor_method	Show a summary of the number of factors by optimization method (e.g., BIC, VSS complexity, Velicer's MAP)
efa_plot	show explained variance by number of factor plot. default is TRUE.
digits	number of digits to round to
n_factor	number of factors for EFA. It will bypass the initial optimization algorithm, and fit the EFA model using this specified number of factor
post_hoc_cfa	a CFA model based on the extracted factor
quite	suppress printing output
streamline	print streamlined output
return_result	If it is set to TRUE (default is FALSE), it will return a fa object from psych

Value

a fa object from psych

Examples

```
efa_summary(lavaan::HolzingerSwineford1939, starts_with("x"), post_hoc_cfa = TRUE)
```

---

get_interaction_term *get interaction term*

Description

get interaction term

Usage

```
get_interaction_term(model)
```

Arguments

model	model

Value

a list with predict vars names

get_predict_df *get factor df to combine with mean_df*

Description

get factor df to combine with mean_df

Usage

```
get_predict_df(data)
```

Arguments

data	data

Value

factor_df

glm_model *Generalized Linear Regression*

Description

[Experimental]
Fit a generalized linear regression using `glm()`. This function is still in early development stage.

Usage

```
glm_model(
  data,
  response_variable,
  predictor_variable,
  two_way_interaction_factor = NULL,
  three_way_interaction_factor = NULL,
  family,
  quite = FALSE
)
```

Arguments

data	data.frame
response_variable	response variable. Support dplyr::select() syntax.
predictor_variable	predictor variable. Support dplyr::select() syntax.
two_way_interaction_factor	two-way interaction factors. You need to pass 2+ factor. Support dplyr::select() syntax.
three_way_interaction_factor	three-way interaction factor. You need to pass exactly 3 factors. Specifying three-way interaction factors automatically included all two-way interactions, so please do not specify the two_way_interaction_factor argument. Support dplyr::select() syntax.
family	a GLM family. It will passed to the family argument in glm See ?glm for possible options.
quite	suppress printing output

Value

an object class of glm representing the linear regression fit

Examples

```
fit <- glm_model(
  response_variable = incidence,
  predictor_variable = period,
  family = "poisson", # or you can enter as poisson(link = 'log'),
  data = lme4::cbpp
)
```

---

glm_model_explore *Exploratory Linear Regression Model Table*

Description

[Experimental]
Exploratory analyses for linear regression models with multiple response, predictor, and two-way interaction variables. (lm models). At the moment, multi-categorical variables are not supported as predictors or interactions (but control is fine). Binary variable should be numeric instead of factor

Usage

```
glm_model_explore(
  data,
  response_variable,
  predictor_variable,
  family,
  two_way_interaction_variable = NULL,
  three_way_interaction_variable = NULL,
  control_variable = NULL,
  marginal_alpha = 0.1,
  verbose = TRUE,
  show_p = TRUE,
  return_result = FALSE,
  print_control = FALSE,
  plot_interaction = FALSE,
  file_dir = NULL,
  device = "jpeg",
  width = 8.5,
  height = 5,
  units = "in",
  y_lim = c(0, 1)
)
```

Arguments

data	data.frame
response_variable	Response variable. Support dplyr::select() syntax.
predictor_variable	Pred variable. Support dplyr::select() syntax.
family	a GLM family. It will passed to the family argument in glm See ?glm for possible options.
two_way_interaction_variable	Two-way interaction variable. Each two-way interaction variable will interact with each pred variable. Support dplyr::select() syntax.
three_way_interaction_variable	Three-way interaction variable. Each three-way interaction variable will interact with each pred and two-way interaction variables. Support dplyr::select() syntax.
control_variable	Control variables. Support dplyr::select() syntax.
marginal_alpha	Set marginal_alpha level for marginally significant (denoted by .). Set to 0.05 if do not want marginally significant denotation.
verbose	Default is TRUE. Set to FALSE to suppress outputs
show_p	Default is TRUE. When TRUE, show the p-value in parenthesis.
return_result	Default is FALSE. If TRUE, it returns the model estimates as a data frame.

print_control	Default is FALSE. If TRUE, print coefficients of control variables.
plot_interaction	Default is FALSE. If TRUE, interactions will be plotted and saved on your laptop.
file_dir	Path of the directory to save plot to
device	Default is "jpeg". See ggplot2::ggsave() for all options.
width	Default is 8.5 (i.e., letter size width).
height	Default is 5.
units	Default is inches. Options are "in", "cm", "mm" or "px".
y_lim	the plot's upper and lower limit for the y-axis. Length of 2. Example: c(lower_limit, upper_limit)

Value

data.frame

Examples

```
test = data.frame(y1 = sample(c(0, 1), size = 1000, replace = TRUE),
y2 = sample(c(0, 1), size = 1000, replace = TRUE),
y3 = sample(c(0, 1), size = 1000, replace = TRUE),
x1 = rnorm(1000,100,10),
x2 = rnorm(1000,10,1),
x3 = rnorm(1000,6,2),
m1 = rnorm(1000,3,1),
m2 = rnorm(1000,2,0.5),
m3 = rnorm(1000,9,0.1),
c1 = rnorm(1000,5,0.4),
c2 = rnorm(1000,2,0.2),
c3 = rnorm(1000,7,0.9)
)

glm_model_explore(data = test,
                  response_variable = c(y1,y2,y3),
                  predictor_variable = c(x1,x2,x3),
                  two_way_interaction_variable = c(m1,m2,m3),
                  family = binomial(link = 'logit'),
                  control_variable = c(c1,c2,c3))
```

---

html_to_pdf *Convert HTML to PDF*

Description

[Experimental]
This is a helper function for knitting Rmd. Due to technological limitation, the output cannot knit to PDF in Rmd directly (the problem is with the latex engine printing unicode character). Therefore, to bypass this problem, you will first need to knit to html file first, then use this function to convert it to a PDF file.

Usage

```
html_to_pdf(file_path = NULL, dir = NULL, scale = 1, render_exist = FALSE)
```

Arguments

file_path	file path to the HTML file (can be relative if you are in a R project)
dir	file path to the directory of all HTML files (can be relative if you are in a R project)
scale	the scale of the PDF
render_exist	overwrite exist PDF. Default is FALSE

Value

no return value

Examples

```
## Not run:
html_to_pdf(file_path = "html_name.html")
# all HTML files in the my_html_folder will be converted
html_to_pdf(dir = "Users/Desktop/my_html_folder")

## End(Not run)
```

---

interaction_plot *Interaction plot*

Description

[Stable]
The function creates interaction plot. By default, it will create an interaction plot using -1 SD and +1 SD of continuous variables, or the two levels of binary variables.

Usage

```
interaction_plot(
  model,
  interaction_term = NULL,
  response_var_name = NULL,
  predict_var1_name = NULL,
  predict_var2_name = NULL,
  predict_var3_name = NULL,
  predict_var1_level = NULL,
  predict_var2_level = NULL,
  predict_var3_level = NULL,
  predict_var1_level_name = NULL,
  predict_var2_level_name = NULL,
```

```
  predict_var3_level_name = NULL,
  y_lim = NULL,
  plot_color = FALSE,
  return_plot_data = FALSE,
  return_plot = FALSE,
  verbose = TRUE,
  print_plot = TRUE,
  data = NULL
)
```

Arguments

`model`	a regression model object from effect.
`interaction_term`	default is the first highest order interaction term in the model. The term should be given explicitly if you want to plot other interaction terms.
`response_var_name`	The name of the response variable can be changed using this setting.
`predict_var1_name`	The name of the first predictor can be changed using this setting.
`predict_var2_name`	The name of the second predictor can be changed using this setting.
`predict_var3_name`	The name of the third predictor can be changed using this setting.
`predict_var1_level`	The default is -1 SD and +1 SD for a continuous variable, and it is the two levels for a binary variable. These can be changed using this setting.
`predict_var2_level`	The default is -1 SD and +1 SD for a continuous variable, and it is the two levels for a binary variable. These can be changed using this setting.
`predict_var3_level`	The default is -1 SD and +1 SD for a continuous variable, and it is the two levels for a binary variable. These can be changed using this setting.
`predict_var1_level_name`	The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the left to right on the x-axis.
`predict_var2_level_name`	The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the top to down on the legend.
`predict_var3_level_name`	The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the left to right on the facets.
`y_lim`	the plot's upper and lower limit for the y-axis. Length of 2. Example: `c(lower_limit, upper_limit)`
`plot_color`	default if `FALSE`. Set to `TRUE` if you want to plot in color

return_plot_data
: default is FALSE. Set to TRUE to return the plot data.

return_plot
: default is FALSE. Set to TRUE to return the plot.

verbose
: deafult is TRUE.

print_plot
: default is TRUE. Set to TRUE to print the plot.

data
: Optional data.frame. Only used when it is not possible to extract data from the model object.

Value

a ggplot object

Examples

```
model_1 <- lm(Sepal.Length ~ Petal.Width * Sepal.Width,
              data = iris)
interaction_plot(model_1)

model_2 <- lm(Sepal.Length ~ Petal.Width * Sepal.Width * Petal.Length,
              data = iris
)

interaction_plot(model_2, # it will automatically select the first three-way interaction term

                 # change the name of the variables of the plot
                 response_var_name = 'SEPAL LENGTH',
                 predict_var1_name = 'PETAL WIDTH',
                 predict_var2_name = 'SEPAL WIDTH',
                 predict_var3_name = 'PETAL LENGTH',

                 # change the number of levels of the variables (e.g., adding the mean)
                 predict_var1_level = c(0.43, 1.19,1.96),
                 predict_var2_level = c(2.62, 3.05,3.49),
                 predict_var3_level = c(1.99,3.758,5.52),
                 predict_var1_level_name = c('-1 SD','Mean','+1 SD'),
                 predict_var2_level_name = c('-1 SD','Mean','+1 SD'),
                 predict_var3_level_name = c('-1 SD','Mean','+1 SD'))
```

---

knit_to_Rmd *Knit Rmd Files Instruction*

Description

This is a helper function that instruct users of the package how to knit a R Markdown (Rmd) files

Usage

```
knit_to_Rmd()
```

Value

no return value

Examples

```
knit_to_Rmd()
```

`label_name` *get label name*

Description

get label name

Usage

```
label_name(
  graph_label_name,
  response_var_name,
  predict_var1_name,
  predict_var2_name,
  predict_var3_name
)
```

Arguments

`graph_label_name`
: label name

`response_var_name`
: outcome variable name

`predict_var1_name`
: predictor 1 name

`predict_var2_name`
: predictor 2 name

`predict_var3_name`
: predictor 3 name

Value

vector of var name

`lme_model_explore` *Exploratory Linear Mixed Effect Model Table*

Description

[Experimental]
Exploratory analyses for linear regression models with multiple response, predictor, and two-way interaction variables. (`lmer` models). At the moment, multi-categorical variables are not supported as predictors or interactions (but control is fine). Binary variable should be `numeric` instead of `factor` This function also do not supports changing random slopes.

Usage

```
lme_model_explore(
  ...,
  data,
  response_variable,
  predictor_variable,
  two_way_interaction_variable = NULL,
  three_way_interaction_variable = NULL,
  random_effect,
  control_variable = NULL,
  marginal_alpha = 0.1,
  return_result = FALSE,
  print_control = FALSE,
  verbose = TRUE,
  show_p = TRUE,
  show_formula = FALSE
)
```

Arguments

`...`	additional parameters pass to lme4::lmer()
`data`	`data.frame`
`response_variable`	Response variable. Support `dplyr::select()` syntax.
`predictor_variable`	Pred variable. Support `dplyr::select()` syntax.
`two_way_interaction_variable`	Two-way interaction variable. Each two-way interaction variable will interact with each pred variable. Support `dplyr::select()` syntax.
`three_way_interaction_variable`	Three-way interaction variable. Each three-way interaction variable will interact with each pred and two-way interaction variables. Support `dplyr::select()` syntax.
`random_effect`	The random-effects terms in the format of `(|)`. See lm4::lmer for specifics.

control_variable
: Control variables. Support `dplyr::select()` syntax.

marginal_alpha
: Set marginal_alpha level for marginally significant (denoted by `.`). Set to 0.05 if do not want marginally significant denotation.

return_result
: Default is `FALSE`. If `TRUE`, it returns the model estimates as a data frame.

print_control
: Default is `FALSE`. If `TRUE`, print coefficients of control variables.

verbose
: Default is `TRUE`. Set to `FALSE` to suppress outputs

show_p
: Default is `TRUE`. When `TRUE`, show the p-value in parenthesis.

show_formula
: Default is `FALSE`. Set to `TRUE` to show the formula.

Value

data.frame

Examples

```
lme_model_explore(data = popular,
                  response_variable = c(popular,extrav),
                  predictor_variable = c(texp),
                  two_way_interaction_variable = sex,
                  random_effect = '(1 | class)')
```

---

lme_model_table *Linear Mixed Effect Model Table*

Description

[Experimental]
Generate tables with multiple response, predictor, or two-way interaction variables (only `lmer` models are supported). You can pass multiple variables for one type of variable (either response, pred, or interaction) only. If you want to pass multiple variables for multiple type of variable, try lmer_model_explore instead. At the moment, multi-categorical variables are not supported as predictors or interactions (but control is fine). Binary variable should be `numeric` instead of `factor` This function also do not supports changing random slopes. Please use `other_parameters` if you want to add non-changing interaction term.

Usage

```
lme_model_table(
  ...,
  data,
  response_variable,
  predictor_variable,
  two_way_interaction_variable = NULL,
```

```
  random_effect,
  control_variable = NULL,
  other_parameters = NULL,
  marginal_alpha = 0.1,
  return_result = FALSE,
  verbose = TRUE,
  show_p = FALSE
)
```

Arguments

...	additional parameters pass to lmerTest::lmer()
data	data.frame
response_variable	response variable. Support dplyr::select() syntax.
predictor_variable	predictor variable. Support dplyr::select() syntax. It will automatically remove the response variable from predictor variable, so you can use contains() or start_with() safely.
two_way_interaction_variable	Two-way interaction variable. Each two-way interaction variable will interact with the predictor variable. Support dplyr::select() syntax.
random_effect	The random-effects terms in the format of (|). See lm4::lmer for specifics.
control_variable	control variables. Support dplyr::select() syntax.
other_parameters	catch call for all other parameters that need to be entered (e.g., non-changing interaction terms). Have to be character type.
marginal_alpha	the set marginal_alpha level for marginally significant (denoted by .). Set to 0.05 if do not want marginally significant denotation.
return_result	It set to TRUE, it return the model estimates data frame.
verbose	default is TRUE. Set to FALSE to suppress outputs
show_p	show the p-value in parenthesis

Value

data.frame

Examples

```
# If you want all varibles to be changing, try lmer_model_explore.
# For more examples, see ?lm_model_table.

# Changing interaction terms with a non-changing response variable
lme_model_table(data = popular,
                response_variable = popular,
                predictor_variable = texp,
```

```
                two_way_interaction_variable = c(extrav,sex),
                random_effect = '(1 | class)')

# A non-changing interaction term with changing response variables
lme_model_table(data = popular,
                response_variable = c(popular,sex),
                predictor_variable = texp,
                other_parameters = 'texp*extrav',
                random_effect = '(1 | class)')
```

`lm_model` *Linear Regressions / ANOVA / ANCOVA*

Description

[Superseded]
Fit a linear regression using `lm()`. Linear regression is used to explore the effect of continuous variables / categorical variables in predicting a normally-distributed continuous variables.

Usage

```
lm_model(
  data,
  response_variable,
  predictor_variable,
  two_way_interaction_factor = NULL,
  three_way_interaction_factor = NULL,
  quite = FALSE
)
```

Arguments

`data`	`data.frame`
`response_variable`	response variable. Support `dplyr::select()` syntax.
`predictor_variable`	predictor variable. Support `dplyr::select()` syntax. It will automatically remove the response variable from predictor variable, so you can use `contains()` or `start_with()` safely.
`two_way_interaction_factor`	two-way interaction factors. You need to pass 2+ factor. Support `dplyr::select()` syntax.
`three_way_interaction_factor`	three-way interaction factor. You need to pass exactly 3 factors. Specifying three-way interaction factors automatically included all two-way interactions, so please do not specify the two_way_interaction_factor argument. Support `dplyr::select()` syntax.
`quite`	suppress printing output

Value

an object class of `lm` representing the linear regression fit

Examples

```
fit <- lm_model(
  data = iris,
  response_variable = Sepal.Length,
  predictor_variable = dplyr::everything(),
  two_way_interaction_factor = c(Sepal.Width, Species)
)
```

---

lm_model_explore *Exploratory Linear Regression Model Table*

Description

[Experimental]
Exploratory analyses for linear regression models with multiple response, predictor, and two-way interaction variables. (`lm` models). At the moment, multi-categorical variables are not supported as predictors or interactions (but control is fine). Binary variable should be `numeric` instead of `factor`

Usage

```
lm_model_explore(
  data,
  response_variable,
  predictor_variable,
  two_way_interaction_variable = NULL,
  three_way_interaction_variable = NULL,
  control_variable = NULL,
  marginal_alpha = 0.1,
  verbose = TRUE,
  show_p = TRUE,
  return_result = FALSE,
  print_control = FALSE,
  plot_interaction = FALSE,
  file_dir = NULL,
  device = "jpeg",
  width = 8.5,
  height = 5,
  units = "in",
  y_lim = NULL
)
```

Arguments

data	data.frame
response_variable	Response variable. Support dplyr::select() syntax.
predictor_variable	Pred variable. Support dplyr::select() syntax.
two_way_interaction_variable	Two-way interaction variable. Each two-way interaction variable will interact with each pred variable. Support dplyr::select() syntax.
three_way_interaction_variable	Three-way interaction variable. Each three-way interaction variable will interact with each pred and two-way interaction variables. Support dplyr::select() syntax.
control_variable	Control variables. Support dplyr::select() syntax.
marginal_alpha	Set marginal_alpha level for marginally significant (denoted by .). Set to 0.05 if do not want marginally significant denotation.
verbose	Default is TRUE. Set to FALSE to suppress outputs
show_p	Default is TRUE. When TRUE, show the p-value in parenthesis.
return_result	Default is FALSE. If TRUE, it returns the model estimates as a data frame.
print_control	Default is FALSE. If TRUE, print coefficients of control variables.
plot_interaction	Default is FALSE. If TRUE, interactions will be plotted and saved on your laptop.
file_dir	Path of the directory to save plot to
device	Default is "jpeg". See ggplot2::ggsave() for all options.
width	Default is 8.5 (i.e., letter size width).
height	Default is 5.
units	Default is inches. Options are "in", "cm", "mm" or "px".
y_lim	the plot's upper and lower limit for the y-axis. Length of 2. Example: c(lower_limit, upper_limit)

Value

data.frame

Examples

```
test = data.frame(y1 = rnorm(1000,2,3),
y2 = rnorm(1000,10,2),
y3 = rnorm(1000,1,4),
x1 = rnorm(1000,100,10),
x2 = rnorm(1000,10,1),
x3 = rnorm(1000,6,2),
m1 = rnorm(1000,3,1),
m2 = rnorm(1000,2,0.5),
```

```
m3 = rnorm(1000,9,0.1),
c1 = rnorm(1000,5,0.4),
c2 = rnorm(1000,2,0.2),
c3 = rnorm(1000,7,0.9)
)


lm_model_explore(data = test,
                 response_variable = c(y1,y2,y3),
                 predictor_variable = c(x1,x2,x3),
                 two_way_interaction_variable = c(m1,m2,m3),
                 control_variable = c(c1,c2,c3))
```

`lm_model_summary` *Model Summary for Linear Regression*

Description

[Superseded]
An integrated function for fitting a linear regression model. This function will no longer be updated. Please use the these functions separately instead: `model_summary`, `interaction_plot`,and `simple_slope`.

Usage

```
lm_model_summary(
  data,
  response_variable = NULL,
  predictor_variable = NULL,
  two_way_interaction_factor = NULL,
  three_way_interaction_factor = NULL,
  family = NULL,
  cateogrical_var = NULL,
  graph_label_name = NULL,
  model_summary = TRUE,
  interaction_plot = TRUE,
  y_lim = NULL,
  plot_color = FALSE,
  digits = 3,
  simple_slope = FALSE,
  assumption_plot = FALSE,
  quite = FALSE,
  streamline = FALSE,
  return_result = FALSE
)
```

Arguments

data	data.frame
response_variable	DV (i.e., outcome variable / response variable). Length of 1. Support dplyr::select() syntax.
predictor_variable	IV. Support dplyr::select() syntax.
two_way_interaction_factor	two-way interaction factors. You need to pass 2+ factor. Support dplyr::select() syntax.
three_way_interaction_factor	three-way interaction factor. You need to pass exactly 3 factors. Specifying three-way interaction factors automatically included all two-way interactions, so please do not specify the two_way_interaction_factor argument. Support dplyr::select() syntax.
family	a GLM family. It will passed to the family argument in glm. See ?glm for possible options. **[Experimental]**
cateogrical_var	list. Specify the upper bound and lower bound directly instead of using ± 1 SD from the mean. Passed in the form of list(var_name1 = c(upper_bound1, lower_bound1),var_name2 = c(upper_bound2, lower_bound2))
graph_label_name	optional vector or function. vector of length 2 for two-way interaction graph. vector of length 3 for three-way interaction graph. Vector should be passed in the form of c(response_var, predict_var1, predict_var2, ...). Function should be passed as a switch function (see ?two_way_interaction_plot for an example)
model_summary	print model summary. Required to be TRUE if you want assumption_plot.
interaction_plot	generate the interaction plot. Default is TRUE
y_lim	the plot's upper and lower limit for the y-axis. Length of 2. Example: c(lower_limit, upper_limit)
plot_color	If it is set to TRUE (default is FALSE), the interaction plot will plot with color.
digits	number of digits to round to
simple_slope	Slope estimate at +1/-1 SD and the mean of the moderator. Uses interactions::sim_slope() in the background.
assumption_plot	Generate an panel of plots that check major assumptions. It is usually recommended to inspect model assumption violation visually. In the background, it calls performance::check_model()
quite	suppress printing output
streamline	print streamlined output
return_result	If it is set to TRUE (default is FALSE), it will return the model, model_summary, and plot (if the interaction term is included)

Value

a list of all requested items in the order of model, model_summary, interaction_plot, simple_slope

Examples

```
fit <- lm_model_summary(
  data = iris,
  response_variable = "Sepal.Length",
  predictor_variable = dplyr::everything(),
  two_way_interaction_factor = c(Sepal.Width, Species),
  interaction_plot = FALSE, # you can also request the interaction plot
  simple_slope = FALSE, # you can also request simple slope estimate
  assumption_plot = FALSE, # you can also request assumption plot
  streamline = FALSE #you can change this to get the least amount of info
)
```

---

lm_model_table *Linear Regression Model Table*

Description

[Experimental]
Generate tables with multiple response, predictor, or two-way interaction variables (only `lm` models are supported). You can pass multiple variables for one type of variable (either response, pred, or interaction) only. If you want to pass multiple variables for multiple type of variable, try lm_model_explore instead. At the moment, multi-categorical variables are not supported as predictors or interactions (but control is fine). Binary variable should be `numeric` instead of `factor`

Usage

```
lm_model_table(
  data,
  response_variable,
  predictor_variable,
  two_way_interaction_variable = NULL,
  control_variable = NULL,
  other_parameters = NULL,
  marginal_alpha = 0.1,
  return_result = FALSE,
  verbose = TRUE,
  show_p = FALSE
)
```

Arguments

`data` data.frame

`response_variable`
response variable. Support `dplyr::select()` syntax.

predictor_variable
: predictor variable. Support `dplyr::select()` syntax. It will automatically remove the response variable from predictor variable, so you can use `contains()` or `start_with()` safely.

two_way_interaction_variable
: Two-way interaction variable. Each two-way interaction variable will interact with the predictor variable. Support `dplyr::select()` syntax.

control_variable
: control variables. Support `dplyr::select()` syntax.

other_parameters
: catch call for all other parameters that need to be entered (e.g., non-changing interaction terms). Have to be `character` type.

marginal_alpha
: the set marginal_alpha level for marginally significant (denoted by .). Set to 0.05 if do not want marginally significant denotation.

return_result
: It set to `TRUE`, it return the model estimates data frame.

verbose
: default is `TRUE`. Set to `FALSE` to suppress outputs

show_p
: show the p-value in parenthesis

Value

data.frame

Examples

```
# If you want all varibles to be changing, try lm_model_explore.

test = data.frame(y1 = rnorm(1000,2,3),
y2 = rnorm(1000,10,2),
y3 = rnorm(1000,1,4),
x1 = rnorm(1000,100,10),
x2 = rnorm(1000,10,1),
x3 = rnorm(1000,6,2),
m1 = rnorm(1000,3,1),
m2 = rnorm(1000,2,0.5),
m3 = rnorm(1000,9,0.1),
c1 = rnorm(1000,5,0.4),
c2 = rnorm(1000,2,0.2),
c3 = rnorm(1000,7,0.9)
)

# Changing response variable
lm_model_table(data = test,
               response_variable = c(y1,y2,y3),
               predictor_variable = x1,
               control_variable = c(c1,c2,c3))

# Changing predictors
lm_model_table(data = test,
               response_variable = y1,
```

```
                predictor_variable = c(x1,x2,x3),
                control_variable = c(c1,c2,c3))


# Changing interaction terms with a non-changing response variable
lm_model_table(data = test,
                response_variable = y1,
                predictor_variable = x1,
                two_way_interaction_variable = c(m1,m2,m3),
                control_variable = c(c1,c2,c3))

# A non-changing interaction term with changing response variables
lm_model_table(data = test,
                response_variable = c(y1,y2,y3),
                predictor_variable = x1,
                other_parameters = c('x1*m1'),
                control_variable = c(c1,c2,c3))
```

`measurement_invariance` *Measurement Invariance*

Description

[Stable]
Compute the measurement invariance model (i.e., measurement equivalence model) using multi-group confirmatory factor analysis (MGCFA; Jöreskog, 1971). This function uses the lavaan::cfa() in the backend. Users can run the configural-metric or the configural-metric-scalar comparisons (see below for detail instruction). All arguments (except the CFA items) must be explicitly named (like model = your-model; see example for inappropriate behavior).

Usage

```
measurement_invariance(
  data,
  ...,
  model = NULL,
  group,
  ordered = FALSE,
  group_partial = NULL,
  invariance_level = "scalar",
  estimator = "ML",
  digits = 3,
  quite = FALSE,
  streamline = FALSE,
  return_result = FALSE
)
```

Arguments

data	data.frame
...	CFA items. Multi-factor CFA items should be separated by comma (as different argument). See below for examples. Support dplyr::select() syntax.
model	explicit lavaan model. Must be specify with model = lavaan_model_syntax. **[Experimental]**
group	the nested variable for multilevel dataset (e.g., Country). Support dplyr::select() syntax.
ordered	Default is FALSE. If it is set to TRUE, lavaan will treat it as a ordinal variable and use DWLS instead of ML
group_partial	items for partial equivalence. The form should be c('DV =~ item1', 'DV =~ item2'). See details for recommended practice.
invariance_level	"metric" or "scalar". Default is 'metric'. Set as 'metric' for configural-metric comparison, and set as 'scalar' for configural-metric-scalar comparison.
estimator	estimator for lavaan. Default is ML
digits	number of digits to round to
quite	suppress printing output except the model summary.
streamline	print streamlined output
return_result	If it is set to TRUE, it will return a data frame of the fit measure summary

Details

Chen (2007) suggested that change in CFI <= |-0.010| supplemented by RMSEA <= 0.015 indicate non-invariance when sample sizes were equal across groups and larger than 300 in each group (Chen, 2007). And, Chen (2007) suggested that change in CFI <= |-0.005| and change in RMSEA <= 0.010 for unequal sample size with each group smaller than 300. For SRMR, Chen (2007) recommend change in SRMR < 0.030 for metric-invariance and change in SRMR < 0.015 for scalar-invariance. For large group size, Rutowski & Svetina (2014) recommended a more liberal cut-off for metric non-invariance for CFI (change in CFI <= |-0.020|) and RMSEA (RMSEA <= 0.030). However, this more liberal cut-off DOES NOT apply to testing scalar non-invariance. If measurement-invariance is not achieved, some researchers suggesting partial invariance is acceptable (by releasing the constraints on some factors). For example, Steenkamp and Baumgartner (1998) suggested that ideally more than half of items on a factor should be invariant. However, it is important to note that no empirical studies were cited to support the partial invariance guideline (Putnick & Bornstein, 2016).

Value

a data.frame of the fit measure summary

References

Chen, F. F. (2007). Sensitivity of Goodness of Fit Indexes to Lack of Measurement Invariance. Structural Equation Modeling: A Multidisciplinary Journal, 14(3), 464–504. https://doi.org/10.1080/10705510701301834

Jöreskog, K. G. (1971). Simultaneous factor analysis in several populations. Psychometrika, 36(4), 409-426.

Putnick, D. L., & Bornstein, M. H. (2016). Measurement Invariance Conventions and Reporting: The State of the Art and Future Directions for Psychological Research. Developmental Review: DR, 41, 71–90. https://doi.org/10.1016/j.dr.2016.06.004

Rutkowski, L., & Svetina, D. (2014). Assessing the Hypothesis of Measurement Invariance in the Context of Large-Scale International Surveys. Educational and Psychological Measurement, 74(1), 31–57. https://doi.org/10.1177/0013164413498257

Steenkamp, J.-B. E. M., & Baumgartner, H. (n.d.). Assessing Measurement Invariance in Cross-National Consumer Research. JOURNAL OF CONSUMER RESEARCH, 13.

Examples

```
# REMEMBER, YOU MUST NAMED ALL ARGUMENT EXCEPT THE CFA ITEMS ARGUMENT
# Fitting a multiple-factor measurement invariance model by passing items.
measurement_invariance(
  x1:x3,
  x4:x6,
  x7:x9,
  data = lavaan::HolzingerSwineford1939,
  group = "school",
  invariance_level = "scalar" # you can change this to metric
)

# Fitting measurement invariance model by passing explicit lavaan model
# I am also going to only test for metric invariance instead of the default scalar invariance

measurement_invariance(
  model = "visual  =~ x1 + x2 + x3;
           textual =~ x4 + x5 + x6;
           speed   =~ x7 + x8 + x9",
  data = lavaan::HolzingerSwineford1939,
  group = "school",
  invariance_level = "metric"
)


## Not run:
# This will fail because I did not add `model = ` in front of the lavaan model.
# Therefore,you must add the tag in front of all arguments
# For example, `return_result = 'model'` instaed of `model`
measurement_invariance(
  "visual  =~ x1 + x2 + x3;
             textual =~ x4 + x5 + x6;
             speed   =~ x7 + x8 + x9",
  data = lavaan::HolzingerSwineford1939
)

## End(Not run)
```

`mediation` *Mediation analysis A Monte Carlo simulation method to assess mediation based on Selig & Preacher (2008).*

Description

Mediation analysis A Monte Carlo simulation method to assess mediation based on Selig & Preacher (2008).

Usage

```
mediation(
  model_med,
  model_y,
  model_med2 = NULL,
  x,
  med,
  med2 = NULL,
  mod = NULL,
  mod_stage = NULL,
  mod_level = NULL,
  conf = 95,
  rep = 20000,
  verbose = TRUE,
  digits = 3
)
```

Arguments

`model_med`	a fitted model object for mediator.
`model_y`	a fitted model object for outcome
`model_med2`	a fitted model object for the second mediator for serial mediation
`x`	a character string indicating the name of the independent variable used in the models.
`med`	a character string indicating the name of the mediator used in the models.
`med2`	a character string indicating the name of the second mediator used in the models (for serial mediations)
`mod`	a character string indicating the name of the moderator used in the models.
`mod_stage`	a character string specifying the stage at which the moderating effect occurs. For instance, in a first-stage moderated mediation, where the moderator influences the effect of X on the mediator (Med), set this to "model_med". In a second-stage moderated mediation, where the moderator affects the relationship between the mediator (Med) and the outcome variable (Y), set this to "model_y".#'
`mod_level`	The default is -1 SD and +1 SD for a continuous variable, and it is the two levels for a binary variable.

conf	level of the returned two-sided confidence intervals. Default is to return the 2.5 and 97.5 percentiles of the simulated quantities (i.e., 95%).
rep	number of Monte Carlo draws
verbose	deafult is TRUE.
digits	number of digits to round to

Value

Nothing to return. Print the indirect effect.

References

Selig, J. P., & Preacher, K. J. (2008, June). Monte Carlo method for assessing mediation: An interactive tool for creating confidence intervals for indirect effects. http://quantpsy.org/.

Examples

```
new_dat = iris %>%
  dplyr::rename(x = Petal.Length) %>%
  dplyr::rename(m = Sepal.Length) %>%
  dplyr::rename(moderator = Sepal.Width) %>%
  dplyr::rename(y = Petal.Width)

model_1 = lm(data = new_dat, m ~ x)
model_2 = lm(data = new_dat, y ~ x*moderator + m)


mediation(model_med = model_1,
          model_y = model_2,
          rep = 20000,
          x = 'x',
          med = 'm',
          mod = 'moderator',
          mod_stage = 'model_y',
          digits = 3)
```

---

mediation_summary *Mediation Analysis*

Description

[Experimental]
It currently only support simple mediation analysis using the path analysis approach with the lavaan package. I am trying to implement multilevel mediation in lavaan. In the future, I will try supporting moderated mediation (through lavaan or mediation) and mediation with latent variable (through lavaan).

Usage

```
mediation_summary(
  data,
  response_variable,
  mediator,
  predictor_variable,
  control_variable = NULL,
  group = NULL,
  standardize = TRUE,
  digits = 3,
  quite = FALSE,
  streamline = FALSE,
  return_result = FALSE
)
```

Arguments

data	data.frame
response_variable	response variable. Support dplyr::select() syntax.
mediator	mediator. Support dplyr::select() syntax.
predictor_variable	predictor variable. Support dplyr::select() syntax.
control_variable	control variables / covariate. Support dplyr::select() syntax.
group	nesting variable for multilevel mediation. Not confident about the implementation method. **[Experimental]**
standardize	standardized coefficients. Default is TRUE
digits	number of digits to round to
quite	suppress printing output
streamline	print streamlined output
return_result	If it is set to TRUE, it will return the lavaan object

Value

an object from lavaan

Examples

```
mediation_summary(
  data = lmerTest::carrots,
  response_variable = Preference,
  mediator = Sweetness,
  predictor_variable = Crisp
)
```

model_summary *Model Summary for Regression Models*

Description

[Stable]
The function will extract the relevant coefficients from the regression models (see below for supported model).

Usage

```
model_summary(
  model,
  digits = 3,
  assumption_plot = FALSE,
  quite = FALSE,
  streamline = TRUE,
  return_result = FALSE,
  standardize = NULL,
  ci_method = "satterthwaite"
)
```

Arguments

model	an model object. The following model are tested for accuracy: lm, glm, lme, lmer, glmer. Other model object may work if it work with parameters::model_parameters()
digits	number of digits to round to
assumption_plot	Generate an panel of plots that check major assumptions. It is usually recommended to inspect model assumption violation visually. In the background, it calls performance::check_model().
quite	suppress printing output
streamline	print streamlined output. Only print model estimate and performance.
return_result	It set to TRUE, it return the model estimates data frame.
standardize	The method used for standardizing the parameters. Can be NULL (default; no standardization), "refit" (for re-fitting the model on standardized data) or one of "basic", "posthoc", "smart", "pseudo". See 'Details' in parameters::standardize_parameters()
ci_method	see options in the Mixed model section in ?parameters::model_parameters()

Value

a list of model estimate data frame, model performance data frame, and the assumption plot (an ggplot object)

References

Nakagawa, S., & Schielzeth, H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution, 4(2), 133–142. https://doi.org/10.1111/j.2041-210x.2012.00261.x

Examples

```
# I am going to show the more generic usage of this function
# You can also use this package's built in function to fit the models
# I recommend using the integrated_multilevel_model_summary to get everything

# lme example
lme_fit <- lme4::lmer("popular ~ texp  + (1 | class)",
  data = popular
)

model_summary(lme_fit)

# lm example

lm_fit <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
  data = iris
)

model_summary(lm_fit)
```

---

polynomial_regression_plot

Polynomial Regression Plot

Description

[Experimental]
The function create a simple regression plot (no interaction). Can be used to visualize polynomial regression.

Usage

```
polynomial_regression_plot(
  model,
  model_data = NULL,
  predictor,
  graph_label_name = NULL,
  x_lim = NULL,
  y_lim = NULL,
  plot_color = FALSE
)
```

Arguments

model	object from lm
model_data	optional dataframe (in case data cannot be retrieved from the model)
predictor	predictor variable name (must be character)
graph_label_name	vector of length 3 or function. Vector should be passed in the form of c(response_var, predict_var1, predict_var2). Function should be passed as a switch function that return the label based on the name passed (e.g., a switch function)
x_lim	the plot's upper and lower limit for the x-axis. Length of 2. Example: c(lower_limit, upper_limit)
y_lim	the plot's upper and lower limit for the y-axis. Length of 2. Example: c(lower_limit, upper_limit)
plot_color	default if FALSE. Set to TRUE if you want to plot in color

Details

It appears that predict cannot handle categorical factors. All variables are converted to numeric before plotting.

Value

an object of class ggplot

Examples

```
fit = lm(data = iris, Sepal.Length ~ poly(Petal.Length,2))
polynomial_regression_plot(model = fit,predictor = 'Petal.Length')
```

---

popular *Popular dataset*

Description

Classic data-set from Chapter 2 of Joop Hox's Multilevel Analysis (2010). The popular dataset included student from different class (i.e., class is the nesting variable). The outcome variable is a self-rated popularity scale. Individual-level (i.e., level 1) predictors are sex, extroversion. Class level (i.e., level 2) predictor is teacher experience.

Usage

```
popular
```

Format

A data frame with 2000 rows and 6 variables:

pupil Subject ID

popular Self-rated popularity scale ranging from 1 to 10

class the class that students belong to (nesting variable)

extrav extraversion scale (individual-level)

sex gender of the student (individual-level)

texp teacher experience (class-level)

Source

http://joophox.net/mlbook2/DataExchange.zip

---

reliability_summary *Reliability Analysis*

Description

[Stable]
First, it will determine whether the data is uni-dimensional or multi-dimensional using `parameters::n_factors()`. If the data is uni-dimensional, then it will print a summary consists of alpha, G6, single-factor CFA, and descriptive statistics result. If it is multi-dimensional, it will print a summary consist of alpha, G6, omega result. You can bypass this by specifying the dimensionality argument.

Usage

```
reliability_summary(
  data,
  cols,
  dimensionality = NULL,
  digits = 3,
  descriptive_table = TRUE,
  quite = FALSE,
  streamline = FALSE,
  return_result = FALSE
)
```

Arguments

`data`	`data.frame`
`cols`	items for reliability analysis. Support `dplyr::select()` syntax.
`dimensionality`	Specify the dimensionality. Either `uni` (uni-dimensionality) or `multi` (multi-dimensionality). Default is `NULL` that determines the dimensionality using EFA.
`digits`	number of digits to round to

descriptive_table	Get descriptive statistics. Default is TRUE
quite	suppress printing output
streamline	print streamlined output
return_result	If it is set to TRUE (default is FALSE), it will return psych::alpha for unidimensional scale, and psych::omega for multidimensional scale.

Value

a psych::alpha object for unidimensional scale, and a psych::omega object for multidimensional scale.

Examples

```
fit <- reliability_summary(data = lavaan::HolzingerSwineford1939, cols = x1:x3)
fit <- reliability_summary(data = lavaan::HolzingerSwineford1939, cols = x1:x9)
```

---

simple_slope *Slope Estimate at Varying Level of Moderators*

Description

[Stable]
The function uses the interaction::sim_slopes() to calculate the slope estimate at varying level of moderators (+/- 1 SD and mean). Additionally, it will produce a Johnson-Newman plot that shows when the slope estimate is not significant

Usage

```
simple_slope(model, data = NULL)
```

Arguments

model	model object from lm, lme,lmer
data	data.frame

Value

a list with the slope estimate data frame and a Johnson-Newman plot.

Examples

```
fit <- lm_model(
  data = iris,
  response_variable = Sepal.Length,
  predictor_variable = dplyr::everything(),
  three_way_interaction_factor = c(Sepal.Width, Petal.Width, Petal.Length)
)

simple_slope_fit <- simple_slope(
  model = fit,
)
```

---

three_way_interaction_plot

Three-way Interaction Plot

Description

[Superseded]
The function creates a three-way interaction plot. By default, it will create an interaction plot with -1 SD and +1 SD of the two continuous variables, or the two levels of the binary variables or dummy-coded multi-categorical variable It has been superseded by `interaction_plot`.

Usage

```
three_way_interaction_plot(
  model,
  interaction_term = NULL,
  response_var_name = NULL,
  predict_var1_name = NULL,
  predict_var2_name = NULL,
  predict_var3_name = NULL,
  predict_var1_level = NULL,
  predict_var2_level = NULL,
  predict_var3_level = NULL,
  predict_var1_level_name = NULL,
  predict_var2_level_name = NULL,
  predict_var3_level_name = NULL,
  y_lim = NULL,
  plot_color = FALSE,
  return_plot_data = FALSE,
  return_plot = FALSE,
  verbose = TRUE,
  print_plot = TRUE,
  data = NULL
)
```

Arguments

`model`	a regression model object from effect.
`interaction_term`	default is the first interaction term in the model. The term should be given explicitly if you want to plot other interaction terms.
`response_var_name`	The name of the response variable can be changed using this setting.
`predict_var1_name`	The name of the first predictor can be changed using this setting.
`predict_var2_name`	The name of the second predictor can be changed using this setting.
`predict_var3_name`	The name of the third predictor can be changed using this setting.
`predict_var1_level`	The default is -1 SD and +1 SD for a continuous variable, and it is the two levels for a binary variable. These can be changed using this setting.
`predict_var2_level`	The default is -1 SD and +1 SD for a continuous variable, and it is the two levels for a binary variable. These can be changed using this setting.
`predict_var3_level`	The default is -1 SD and +1 SD for a continuous variable, and it is the two levels for a binary variable. These can be changed using this setting.
`predict_var1_level_name`	The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the left to right on the x-axis.
`predict_var2_level_name`	The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the top to down on the legend.
`predict_var3_level_name`	The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the left to right on the facets.
`y_lim`	the plot's upper and lower limit for the y-axis. Length of 2. Example: `c(lower_limit, upper_limit)`
`plot_color`	default if `FALSE`. Set to `TRUE` if you want to plot in color
`return_plot_data`	default is `FALSE`. Set to `TRUE` to return the plot data.
`return_plot`	default is `FALSE`. Set to `TRUE` to return the plot.
`verbose`	deafult is `TRUE`.
`print_plot`	default is `TRUE`. Set to `TRUE` to print the plot.
`data`	Optional data.frame. Only used when it is not possible to extract data from the model object.

Value

an object of class `ggplot`

`two_way_interaction_plot` *Two-way Interaction Plot*

Description

[Superseded]
The function creates a three-way interaction plot. By default, it will create an interaction plot with -1 SD and +1 SD of the two continuous variables, or the two levels of the binary variables or dummy-coded multi-categorical variable It has been superseded by `interaction_plot`.

Usage

```
two_way_interaction_plot(
  model,
  interaction_term = NULL,
  response_var_name = NULL,
  predict_var1_name = NULL,
  predict_var2_name = NULL,
  predict_var1_level = NULL,
  predict_var2_level = NULL,
  predict_var1_level_name = NULL,
  predict_var2_level_name = NULL,
  y_lim = NULL,
  plot_color = FALSE,
  return_plot_data = FALSE,
  return_plot = FALSE,
  verbose = FALSE,
  print_plot = TRUE,
  data = NULL
)
```

Arguments

`model`	a regression model object from effect.
`interaction_term`	default is the first interaction term in the model. The term should be given explicitly if you want to plot other interaction terms.
`response_var_name`	The name of the response variable can be changed using this value.
`predict_var1_name`	The name of the first predictor can be changed using this value.
`predict_var2_name`	The name of the second predictor can be changed using this value.
`predict_var1_level`	default is the -1 SD and +1 SD for continuous variable. They can be changed using this value.

predict_var2_level
: default is the -1 SD and +1 SD for continuous variable. They can be changed using this value.

predict_var1_level_name
: The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the left to right on the x-axis.

predict_var2_level_name
: The labels of the level can be change using this value (e.g., `c('-1 SD','+1 SD')`). The order should be from the top to down on the legend.

y_lim
: the plot's upper and lower limit for the y-axis. Length of 2. Example: `c(lower_limit, upper_limit)`

plot_color
: default if `FALSE`. Set to `TRUE` if you want to plot in color

return_plot_data
: default is `FALSE`. Set to `TRUE` to return the plot data.

return_plot
: default is `FALSE`. Set to `TRUE` to return the plot.

verbose
: deafult is `TRUE`.

print_plot
: default is `TRUE`. Set to `TRUE` to print the plot.

data
: Optional data.frame. Only used when it is not possible to extract data from the model object.

Details

It appears that "predict' cannot handle categorical factors. All variables are converted to numeric before plotting.

Value

an object of class `ggplot`

Index

∗ **datasets**
    acitelli, 2
    popular, 43

acitelli, 2
anova_plot, 3
APIM_sem, 4
APIM_table, 6

cfa_groupwise, 7
cfa_summary, 8
compare_fit, 10
cor_test, 11
cronbach_alpha, 13

descriptive_table, 14

efa_summary, 15
effect, *22*, *47, 48*

get_interaction_term, 16
get_predict_df, 17
glm_model, 17
glm_model_explore, 18

html_to_pdf, 20

interaction_plot, 21, *31*, *46*, *48*

knit_to_Rmd, 23

label_name, 24
lm_model, 28
lm_model_explore, 29
lm_model_summary, 31
lm_model_table, 33
lme_model_explore, 25
lme_model_table, 26

measurement_invariance, 35
mediation, 38
mediation_summary, 39
model_summary, *31*, 41

polynomial_regression_plot, 42
popular, 43

reliability_summary, 44

simple_slope, *31*, 45

three_way_interaction_plot, 46
two_way_interaction_plot, 48