

Package ‘psycModel’

July 23, 2025

Type Package

Title Integrated Toolkit for Psychological Analysis and Modeling in R

Version 0.5.0

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

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Language en-US

NeedsCompilation no

Author Jason Moy [aut, cre] (ORCID: <<https://orcid.org/0000-0001-8795-3311>>)

Maintainer Jason Moy <jasonmoy28@gmail.com>

Repository CRAN

Date/Publication 2023-11-01 06:00:02 UTC

Contents

anova_plot	2
cfa_groupwise	3
cfa_summary	4
compare_fit	7
cor_test	8
cronbach_alpha	9
descriptive_table	10
efa_summary	11
get_interaction_term	12
get_predict_df	13
glm_model	13
html_to_pdf	14
interaction_plot	15
knit_to_Rmd	16
label_name	17
lme_model	18
lme_multilevel_model_summary	20
lm_model	23
lm_model_summary	24
lm_model_table	26
measurement_invariance	27
mediation_summary	29
model_summary	31
polynomial_regression_plot	32
popular	33
reliability_summary	34
simple_slope	35
three_way_interaction_plot	36
two_way_interaction_plot	37
Index	39

anova_plot	<i>ANOVA Plot</i>
------------	-------------------

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)
```

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)

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)
```

cfa_groupwise

Confirmatory Factor Analysis (groupwise)

Description**[Stable]**

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 <code>dplyr::select()</code> syntax.
group	character. group variable. Support <code>dplyr::select()</code> syntax.
model	explicit lavaan model. Must be specify with <code>model = lavaan_model_syntax</code> . [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	<i>Confirmatory Factor Analysis</i>
-------------	-------------------------------------

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,
  quiet = FALSE,
  return_result = FALSE
)
```

Arguments

<code>data</code>	data frame
<code>...</code>	CFA items. Multi-factor CFA items should be separated by comma (as different argument). See below for examples. Support <code>dplyr::select()</code> syntax.
<code>model</code>	explicit lavaan model. Must be specify with <code>model = lavaan_model_syntax</code> . [Experimental]
<code>group</code>	optional character. used for multi-level CFA. the nested variable for multilevel dataset (e.g., Country). Support <code>dplyr::select()</code> syntax.
<code>ordered</code>	Default is FALSE. If it is set to TRUE, lavaan will treat it as a ordinal variable and use DWLS instead of ML
<code>digits</code>	number of digits to round to
<code>estimator</code>	estimator for lavaan. Default is ML
<code>model_covariance</code>	print model covariance. Default is TRUE
<code>model_variance</code>	print model variance. Default is TRUE
<code>plot</code>	print a path diagram. Default is TRUE
<code>group_partial</code>	Items for partial equivalence. The form should be <code>c('DV =~ item1', 'DV =~ item2')</code> .
<code>streamline</code>	print streamlined output
<code>quiet</code>	suppress printing output
<code>return_result</code>	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 indices (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/10.1080/1070551990954011>

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,
  quiet = 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'` instead 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,
  quiet = 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 <code>performance::compare_performance()</code> .
digits	number of digits to round to
quiet	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,
  method = "pearson",
  p_adjust = "none",
  streamline = FALSE,
  quiet = FALSE,
  return_result = FALSE
)
```

Arguments

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

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 separately
- 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	<i>Descriptive Statistics Table</i>
-------------------	-------------------------------------

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,
  quiet = FALSE,
  return_result = FALSE
)
```

Arguments

<code>data</code>	<code>data.frame</code>
<code>cols</code>	column(s) need to be included in the table. Support <code>dplyr::select()</code> syntax.
<code>...</code>	additional arguments passed to <code>cor_test</code> . See <code>?cor_test</code> .
<code>digits</code>	number of digit for the descriptive table
<code>descriptive_indicator</code>	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)
<code>file_path</code>	file path for export. The function will implicitly pass this argument to the <code>write.csv(file = file_path)</code>

streamline print streamlined output
 quite suppress printing output
 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

<code>data</code>	<code>data.frame</code>
<code>cols</code>	columns. Support <code>dplyr::select()</code> syntax.
<code>rotation</code>	the rotation to use in estimation. Default is 'oblimin'. Options are 'none', 'varimax', 'quartimax', 'promax', 'oblimin', or 'simplimax'
<code>optimal_factor_method</code>	Show a summary of the number of factors by optimization method (e.g., BIC, VSS complexity, Velicer's MAP)
<code>efa_plot</code>	show explained variance by number of factor plot. default is TRUE.
<code>digits</code>	number of digits to round to
<code>n_factor</code>	number of factors for EFA. It will bypass the initial optimization algorithm, and fit the EFA model using this specified number of factor
<code>post_hoc_cfa</code>	a CFA model based on the extracted factor
<code>quite</code>	suppress printing output
<code>streamline</code>	print streamlined output
<code>return_result</code>	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

<code>model</code>	<code>model</code>
--------------------	--------------------

Value

a list with predict vars names

get_predict_df	<i>get factor df to combine with mean_df</i>
----------------	--

Description

get factor df to combine with mean_df

Usage

```
get_predict_df(data)
```

Arguments

data	data
------	------

Value

factor_df

glm_model	<i>Generalized Linear Regression</i>
-----------	--------------------------------------

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,  
  quiet = FALSE  
)
```

Arguments

<code>data</code>	<code>data.frame</code>
<code>response_variable</code>	response variable. Support <code>dplyr::select()</code> syntax.
<code>predictor_variable</code>	predictor variable. Support <code>dplyr::select()</code> syntax.
<code>two_way_interaction_factor</code>	two-way interaction factors. You need to pass 2+ factor. Support <code>dplyr::select()</code> syntax.
<code>three_way_interaction_factor</code>	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 <code>two_way_interaction_factor</code> argument. Support <code>dplyr::select()</code> syntax.
<code>family</code>	a GLM family. It will passed to the <code>family</code> argument in <code>glmer</code> . See <code>?glmer</code> for possible options.
<code>quite</code>	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
)
```

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	<i>Interaction plot</i>
------------------	-------------------------

Description**[Stable]**

The function creates a two-way or three-way interaction plot. It will creates a plot with ± 1 SD from the mean of the independent variable. See below for supported model. I recommend using concurrently with `lm_model()`, `lme_model()`.

Usage

```
interaction_plot(
  model,
  data = NULL,
  graph_label_name = NULL,
  cateogrical_var = NULL,
  y_lim = NULL,
  plot_color = FALSE
)
```

Arguments

model	object from lme, lme4, lmerTest object.
data	data frame. If the function is unable to extract data frame from the object, then you may need to pass it directly
graph_label_name	vector of length 4 or a switch function (see ?two_way_interaction_plot example). Vector should be passed in the form of c(response_var, predict_var1, predict_var2, predict_var3).
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))
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

Value

a ggplot object

Examples

```
lm_fit_2 <- lm(Sepal.Length ~ Sepal.Width + Petal.Length +
  Sepal.Width*Petal.Length, data = iris)

interaction_plot(lm_fit_2)

lm_fit_3 <- lm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width +
  Sepal.Width*Petal.Length:Petal.Width, data = iris)

interaction_plot(lm_fit_3)
```

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	<i>get label name</i>
------------	-----------------------

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

*Linear Mixed Effect Model***Description****[Stable]**

Fit a linear mixed effect model (i.e., hierarchical linear model, multilevel linear model) using the `nlme::lme()` or the `lmerTest::lmer()` function. Linear mixed effect model is used to explore the effect of continuous / categorical variables in predicting a normally distributed continuous variable.

Usage

```
lme_model(
  data,
  model = NULL,
  response_variable,
  random_effect_factors = NULL,
  non_random_effect_factors = NULL,
  two_way_interaction_factor = NULL,
  three_way_interaction_factor = NULL,
  id,
  estimation_method = "REML",
  opt_control = "bobyqa",
  na.action = stats::na.omit,
  use_package = "lmerTest",
  quiet = FALSE
)
```

Arguments

<code>data</code>	<code>data.frame</code>
<code>model</code>	lme4 model syntax. Support more complicated model. Note that <code>model_summary</code> will only return fixed effect estimates.
<code>response_variable</code>	DV (i.e., outcome variable / response variable). Length of 1. Support <code>dplyr::select()</code> syntax.
<code>random_effect_factors</code>	random effect factors (level-1 variable for HLM people) Factors that need to estimate fixed effect and random effect (i.e., random slope / varying slope based on the id). Support <code>dplyr::select()</code> syntax.
<code>non_random_effect_factors</code>	non-random effect factors (level-2 variable for HLM people). Factors only need to estimate fixed effect. Support <code>dplyr::select()</code> syntax.
<code>two_way_interaction_factor</code>	two-way interaction factors. You need to pass 2+ factor. Support <code>dplyr::select()</code> 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 <code>dplyr::select()</code> syntax.
id	the nesting variable (e.g. group, time). Length of 1. Support <code>dplyr::select()</code> syntax.
estimation_method	character. ML or REML default to REML.
opt_control	default is <code>optim</code> for <code>lme</code> and <code>bobyqa</code> for <code>lmerTest</code>
na.action	default is <code>stats::na.omit</code> . Another common option is <code>na.exclude</code>
use_package	Default is <code>lmerTest</code> . Only available for linear mixed effect model. Options are <code>nlme</code> , <code>lmerTest</code> , or <code>lme4</code> ('lme4' return similar result as <code>lmerTest</code> except the return model)
quite	suppress printing output

Details

Here is a little tip. If you are using generic selecting syntax (e.g., `contains()` or `start_with()`), you don't need to remove the response variable and the id from the factors. It will be automatically remove. For example, if you have `x1:x9` as your factors. You want to regress `x2:x8` on `x1`. Your probably pass something like `response_variable = x1, random_effect_factors = c(contains('x'),-x1)` to the function. However, you don't need to do that, you can just pass `random_effect_factors = c(contains('x'))` to the function since it will automatically remove the response variable from selection.

Value

an object representing the linear mixed-effects model fit (it maybe an object from `lme` or `lmer` depending of the package you use)

Examples

```
# two-level model with level-1 and level-2 variable with random intercept and random slope
fit1 <- lme_model(
  data = popular,
  response_variable = popular,
  random_effect_factors = c(extrav, sex),
  non_random_effect_factors = texp,
  id = class
)
```

```
# added two-way interaction factor
fit2 <- lme_model(
  data = popular,
  response_variable = popular,
  random_effect_factors = c(extrav, sex),
  non_random_effect_factors = texp,
```

```

    two_way_interaction_factor = c(extrav, texp),
    id = class
  )

  # pass a explicit lme model (I don't why you want to do that, but you can)
  lme_fit <- lme_model(
    model = "popular ~ extrav*texp + (1 + extrav | class)",
    data = popular
  )

```

lme_multilevel_model_summary

Model Summary for Mixed Effect Model

Description

[Stable]

An integrated function for fitting a multilevel linear regression (also known as hierarchical linear regression).

Usage

```

lme_multilevel_model_summary(
  data,
  model = NULL,
  response_variable = NULL,
  random_effect_factors = NULL,
  non_random_effect_factors = NULL,
  two_way_interaction_factor = NULL,
  three_way_interaction_factor = NULL,
  family = NULL,
  cateogrical_var = NULL,
  id = NULL,
  graph_label_name = NULL,
  estimation_method = "REML",
  opt_control = "bobyqa",
  na.action = stats::na.omit,
  model_summary = TRUE,
  interaction_plot = TRUE,
  y_lim = NULL,
  plot_color = FALSE,
  digits = 3,
  use_package = "lmerTest",
  standardize = NULL,
  ci_method = "satterthwaite",
  simple_slope = FALSE,
  assumption_plot = FALSE,
  quite = FALSE,

```

```

    streamline = FALSE,
    return_result = FALSE
  )

```

Arguments

<code>data</code>	<code>data.frame</code>
<code>model</code>	lme4 model syntax. Support more complicated model structure from lme4. It is not well-tested to ensure accuracy [Experimental]
<code>response_variable</code>	DV (i.e., outcome variable / response variable). Length of 1. Support <code>dplyr::select()</code> syntax.
<code>random_effect_factors</code>	random effect factors (level-1 variable for HLM from a HLM perspective) Factors that need to estimate fixed effect and random effect (i.e., random slope / varying slope based on the id). Support <code>dplyr::select()</code> syntax.
<code>non_random_effect_factors</code>	non-random effect factors (level-2 variable from a HLM perspective). Factors only need to estimate fixed effect. Support <code>dplyr::select()</code> syntax.
<code>two_way_interaction_factor</code>	two-way interaction factors. You need to pass 2+ factor. Support <code>dplyr::select()</code> syntax.
<code>three_way_interaction_factor</code>	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 <code>two_way_interaction_factor</code> argument. Support <code>dplyr::select()</code> syntax.
<code>family</code>	a GLM family. It will passed to the family argument in glmer. See ?glmer for possible options. [Experimental]
<code>cateogrical_var</code>	list. Specify the upper bound and lower bound directly instead of using ± 1 SD from the mean. Passed in the form of <code>list(var_name1 = c(upper_bound1, lower_bound1), var_name2 = c(upper_bound2, lower_bound2))</code>
<code>id</code>	the nesting variable (e.g. group, time). Length of 1. Support <code>dplyr::select()</code> syntax.
<code>graph_label_name</code>	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 <code>c(response_var, predict_var1, predict_var2, ...)</code> . Function should be passed as a switch function (see ?two_way_interaction_plot for an example)
<code>estimation_method</code>	character. ML or REML default is REML.
<code>opt_control</code>	default is optim for lme and bobyqa for lmerTest.
<code>na.action</code>	default is stats::na.omit. Another common option is na.exclude
<code>model_summary</code>	print model summary. Required to be TRUE if you want assumption_plot.

interaction_plot	generate 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
use_package	Default is lmerTest. Only available for linear mixed effect model. Options are nlme, lmerTest, or lme4('lme4 return similar result as lmerTest except the return model)
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()
simple_slope	Slope estimate at ± 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 (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 <- lme_multilevel_model_summary(
  data = popular,
  response_variable = popular,
  random_effect_factors = NULL, # you can add random effect predictors here
  non_random_effect_factors = c(extrav, texp),
  two_way_interaction_factor = NULL, # you can add two-way interaction plot here
  graph_label_name = NULL, #you can also change graph lable name here
  id = class,
  simple_slope = FALSE, # you can also request simple slope estimate
  assumption_plot = FALSE, # you can also request assumption plot
  plot_color = FALSE, # you can also request the plot in color
  streamline = FALSE # you can change this to get the least amount of info
)
```

lm_model

Linear Regressions / ANOVA / ANCOVA

Description

[Stable]

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,  
  quiet = FALSE  
)
```

Arguments

<code>data</code>	<code>data.frame</code>
<code>response_variable</code>	response variable. Support <code>dplyr::select()</code> syntax.
<code>predictor_variable</code>	predictor variable. Support <code>dplyr::select()</code> syntax. It will automatically remove the response variable from predictor variable, so you can use <code>contains()</code> or <code>start_with()</code> safely.
<code>two_way_interaction_factor</code>	two-way interaction factors. You need to pass 2+ factor. Support <code>dplyr::select()</code> syntax.
<code>three_way_interaction_factor</code>	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 <code>two_way_interaction_factor</code> argument. Support <code>dplyr::select()</code> syntax.
<code>quiet</code>	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_summary

*Model Summary for Linear Regression***Description****[Stable]**

An integrated function for fitting a linear regression model.

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 <code>dplyr::select()</code> 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 <code>two_way_interaction_factor</code> argument. Support <code>dplyr::select()</code> syntax.
family	a GLM family. It will be passed to the <code>family</code> argument in <code>glm</code> . See <code>?glm</code> for possible options. [Experimental]
categorical_var	list. Specify the upper bound and lower bound directly instead of using ± 1 SD from the mean. Passed in the form of <code>list(var_name1 = c(upper_bound1, lower_bound1), var_name2 = c(upper_bound2, lower_bound2))</code>
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 <code>c(response_var, predict_var1, predict_var2, ...)</code> . Function should be passed as a switch function (see <code>?two_way_interaction_plot</code> for an example)
model_summary	print model summary. Required to be TRUE if you want <code>assumption_plot</code> .
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: <code>c(lower_limit, upper_limit)</code>
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 <code>interactions::sim_slope()</code> 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 <code>performance::check_model()</code>
quiet	suppress printing output
streamline	print streamlined output
return_result	If it is set to TRUE (default is FALSE), it will return the <code>model</code> , <code>model_summary</code> , and <code>plot</code> (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	<i>Linear Regression Model Table Generate tables with multiple response and predictor variable (only lm models are supported)</i>
----------------	---

Description

Linear Regression Model Table Generate tables with multiple response and predictor variable (only lm models are supported)

Usage

```

lm_model_table(
  data,
  response_variable,
  predictor_variable,
  control_variable = 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.
control_variable	control variables. Support dplyr::select() syntax.
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

```
lm_model_table(data = iris,
               response_variable = c(Sepal.Length, Sepal.Width),
               predictor_variable = Petal.Width)
```

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,
  quiet = FALSE,
  streamline = FALSE,
  return_result = FALSE
)
```

Arguments

<code>data</code>	data.frame
<code>...</code>	CFA items. Multi-factor CFA items should be separated by comma (as different argument). See below for examples. Support <code>dplyr::select()</code> syntax.
<code>model</code>	explicit lavaan model. Must be specify with <code>model = lavaan_model_syntax</code> . [Experimental]

group	the nested variable for multilevel dataset (e.g., Country). Support <code>dplyr::select()</code> 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 <code>c('DV =~ item1', 'DV =~ item2')</code> . 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 $\leq | -0.010 |$ 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 $\leq | -0.005 |$ 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 $\leq | -0.020 |$) 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'` instead of `model`
measurement_invariance(
  "visual =~ x1 + x2 + x3;
   textual =~ x4 + x5 + x6;
   speed =~ x7 + x8 + x9",
  data = lavaan::HolzingerSwineford1939
)

## End(Not run)
```

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,
  quiet = 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
quiet	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,
  quiet = 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().
quiet	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 R^2 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)
teexp 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

<code>data</code>	<code>data.frame</code>
<code>cols</code>	items for reliability analysis. Support <code>dplyr::select()</code> syntax.
<code>dimensionality</code>	Specify the dimensionality. Either <code>uni</code> (uni-dimensionality) or <code>multi</code> (multi-dimensionality). Default is <code>NULL</code> that determines the dimensionality using EFA.
<code>digits</code>	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 <code>psych::alpha</code> for unidimensional scale, and <code>psych::omega</code> 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 <code>lm</code> , <code>lme</code> , <code>lmer</code>
data	<code>data.frame</code>

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**[Deprecated]**

The function creates a two-way interaction plot. It will creates a plot with ± 1 SD from the mean of the independent variable. See below for supported model. I recommend using concurrently with `lm_model()`, `lme_model()`.

Usage

```
three_way_interaction_plot(
  model,
  data = NULL,
  cateogrical_var = NULL,
  graph_label_name = NULL,
  y_lim = NULL,
  plot_color = FALSE
)
```

Arguments

<code>model</code>	object from <code>lme</code> , <code>lme4</code> , <code>lmerTest</code> object.
<code>data</code>	<code>data.frame</code> . If the function is unable to extract data frame from the object, then you may need to pass it directly
<code>cateogrical_var</code>	list. Specify the upper bound and lower bound directly instead of using ± 1 SD from the mean. Passed in the form of <code>list(var_name1 = c(upper_bound1, lower_bound1), var_name2 = c(upper_bound2, lower_bound2))</code>
<code>graph_label_name</code>	vector of length 4 or a switch function (see <code>?two_way_interaction_plot</code> example). Vector should be passed in the form of <code>c(response_var, predict_var1, predict_var2, predict_var3)</code> .

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

a ggplot object

Examples

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

three_way_interaction_plot(lm_fit, data = iris)
```

two_way_interaction_plot	<i>Two-way Interaction Plot</i>
--------------------------	---------------------------------

Description

[Deprecated]

The function creates a two-way interaction plot. It will creates a plot with ± 1 SD from the mean of the independent variable. See supported model below. I recommend using concurrently with `lm_model` or `lme_model`.

Usage

```
two_way_interaction_plot(
  model,
  data = NULL,
  graph_label_name = NULL,
  cateogrical_var = NULL,
  y_lim = NULL,
  plot_color = FALSE
)
```

Arguments

<code>model</code>	object from <code>lm</code> , <code>nlme</code> , <code>lme4</code> , or <code>lmerTest</code>
<code>data</code>	<code>data.frame</code> . If the function is unable to extract data frame from the object, then you may need to pass it directly
<code>graph_label_name</code>	vector of length 3 or function. Vector should be passed in the form of <code>c(response_var, predict_var1, predict_var2)</code> . Function should be passed as a switch function that return the label based on the name passed (e.g., a switch function)
<code>cateogrical_var</code>	list. Specify the upper bound and lower bound directly instead of using ± 1 SD from the mean. Passed in the form of <code>list(var_name1 = c(upper_bound1, lower_bound1), var_name2 = c(upper_bound2, lower_bound2))</code>
<code>y_lim</code>	the plot's upper and lower limit for the y-axis. Length of 2. Example: <code>c(lower_limit, upper_limit)</code>
<code>plot_color</code>	default if <code>FALSE</code> . Set to <code>TRUE</code> 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

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

Index

- * **datasets**
 - popular, [33](#)
- anova_plot, [2](#)
- cfa_groupwise, [3](#)
- cfa_summary, [4](#)
- compare_fit, [7](#)
- cor_test, [8](#)
- cronbach_alpha, [9](#)
- descriptive_table, [10](#)
- efa_summary, [11](#)
- get_interaction_term, [12](#)
- get_predict_df, [13](#)
- glm_model, [13](#)
- html_to_pdf, [14](#)
- interaction_plot, [15](#)
- knit_to_Rmd, [16](#)
- label_name, [17](#)
- lm_model, [23](#)
- lm_model_summary, [24](#)
- lm_model_table, [26](#)
- lme_model, [18](#)
- lme_multilevel_model_summary, [20](#)
- measurement_invariance, [27](#)
- mediation_summary, [29](#)
- model_summary, [31](#)
- polynomial_regression_plot, [32](#)
- popular, [33](#)
- reliability_summary, [34](#)
- simple_slope, [35](#)
- three_way_interaction_plot, [36](#)
- two_way_interaction_plot, [37](#)