Package 'guess'

March 31, 2026

Title Adjust Estimates of Learning for Guessing

Version 0.3.0

Description Provides tools to adjust estimates of learning for guessing-related bias in educational and survey research. Implements standard guessing correction methods and a sophisticated latent class model that leverages informative pre-post test transitions to account for guessing behavior. The package helps researchers obtain more accurate estimates of actual learning when respondents may guess on closed-ended knowledge items. For theoretical background and empirical validation, see Cor and Sood (2018) <https://gsood.com/research/papers/guess.pdf>.

URL https://github.com/finite-sample/guess, https://finite-sample.github.io/guess/

BugReports https://github.com/finite-sample/guess/issues

Depends R (>= 4.0.0)

Imports Rsolnp, stats, checkmate

License MIT + file LICENSE

VignetteBuilder knitr

Suggests knitr (>= 1.11), rmarkdown, testthat (>= 3.0.0), lintr, covr

RoxygenNote 7.3.3

Encoding UTF-8

Language en-US

Config/testthat/edition 3

NeedsCompilation no

Author Gaurav Sood [aut, cre], Ken Cor [aut]

Maintainer Gaurav Sood <gsood07@gmail.com>

Repository CRAN

Date/Publication 2026-03-31 04:10:02 UTC

Contents

`calculate_expected_values` *Calculate expected values for goodness of fit test*

Description

Calculate expected values for goodness of fit test

Usage

```
calculate_expected_values(gamma_i, params, total_obs, model_type = "nodk")
```

Arguments

`gamma_i`	item-specific gamma value
`params`	estimated parameters for the item
`total_obs`	total observations for the item
`model_type`	"nodk" or "dk" model

Value

vector of expected values

fit_model *Goodness of fit statistics for transition matrix data*

Description

Chi-square goodness of fit between true and model based multivariate distribution. Handles both data with and without don't know responses automatically.

Usage

```
fit_model(pre_test, pst_test, g, est.param, force9 = FALSE)

fit_dk(pre_test, pst_test, g, est.param, force9 = FALSE)

fit_nodk(pre_test, pst_test, g, est.param)
```

Arguments

pre_test	data.frame carrying pre_test items
pst_test	data.frame carrying pst_test items
g	estimates of gamma produced from lca_cor
est.param	estimated parameters produced from lca_cor
force9	Optional. Force 9-column format even if no DK responses. Default is FALSE.

Details

Unified Goodness of Fit Statistics

Value

matrix with two rows: top row carrying chi-square value, bottom row p-values

Examples

```
## Not run:
# Fit model first
transmatrix <- multi_transmat(pre_test, pst_test)
res <- lca_cor(transmatrix)

# Calculate goodness of fit
fit_stats <- fit_model(pre_test, pst_test, res$param.lca[nrow(res$param.lca), ],
                       res$param.lca[-nrow(res$param.lca), ])

## End(Not run)
```

`format_transition_matrix` *Format transition matrix result with appropriate row and column names*

Description

Format transition matrix result with appropriate row and column names

Usage

```
format_transition_matrix(transition_list, n_items, add_aggregate = FALSE)
```

Arguments

`transition_list`	list of transition vectors
`n_items`	number of items
`add_aggregate`	whether to add aggregate row

Value

formatted matrix

`group_adj` *Group Level Adjustment That Accounts for Propensity to Guess*

Description

Adjusts observed 1s based on propensity to guess (based on observed 0s) and item level γ. You can also put in your best estimate of hidden knowledge behind don't know responses.

Usage

```
group_adj(pre = NULL, pst = NULL, gamma = NULL, dk = 0.03)
```

Arguments

`pre`	pre data frame. Required. Each vector within the data frame should only take values 0, 1, and 'd'.
`pst`	pst data frame. Required. Each vector within the data frame should only take values 0, 1, and 'd'.
`gamma`	probability of getting the right answer without knowledge
`dk`	Numeric. Between 0 and 1. Hidden knowledge behind don't know responses. Default is .03.

Value

nested list of pre and post adjusted responses, and adjusted learning estimates

Examples

```
pre_test_var <- data.frame(pre = c(1,0,0,1,"d","d",0,1,NA))
pst_test_var <- data.frame(pst = c(1,NA,1,"d",1,0,1,1,"d"))
gamma <- c(.25)
group_adj(pre_test_var, pst_test_var, gamma)
```

`lca_adj` *Person Level Adjustment*

Description

Adjusts observed 1s based on item level parameters of the LCA model. Currently only takes data with Don't Know. And treats don't know responses as true confessions on ignorance. If NAs are observed in the data, they are treating as acknowledgments of ignorance.

Usage

```
lca_adj(pre = NULL, pst = NULL)
```

Arguments

`pre`	pre data frame
`pst`	pst data frame

Value

list of pre and post adjusted responses

Examples

```
pre_test_var <- data.frame(pre = c(1, 0, 0, 1, "d", "d", 0, 1, NA))
pst_test_var <- data.frame(pst = c(1, NA, 1, "d", 1, 0, 1, 1, "d"))
lca_adj(pre_test_var, pst_test_var)
```

lca_cor *Calculate item level and aggregate learning*

Description

guesstimate

Usage

```
lca_cor(
  transmatrix = NULL,
  nodk_priors = c(0.3, 0.1, 0.1, 0.25),
  dk_priors = c(0.3, 0.1, 0.2, 0.05, 0.1, 0.1, 0.05, 0.25)
)
```

Arguments

transmatrix	transition matrix returned from multi_transmat
nodk_priors	Optional. Vector of length 4. Priors for the parameters for model that fits data without Don't Knows
dk_priors	Optional. Vector of length 8. Priors for the parameters for model that fits data with Don't Knows

Value

list with two items: parameter estimates and estimates of learning

Examples

```
# Without DK
pre_test <- data.frame(item1 = c(1, 0, 0, 1, 0), item2 = c(1, NA, 0, 1, 0))
pst_test <- pre_test + cbind(c(0, 1, 1, 0, 0), c(0, 1, 0, 0, 1))
transmatrix <- multi_transmat(pre_test, pst_test)
res <- lca_cor(transmatrix)
```

lca_se *Bootstrapped standard errors of effect size estimates*

Description

guess_stnderr

Usage

```
lca_se(
  pre_test = NULL,
  pst_test = NULL,
  nsamps = 100,
  seed = 31415,
  force9 = FALSE
)
```

Arguments

pre_test	data.frame carrying pre_test items
pst_test	data.frame carrying pst_test items
nsamps	number of resamples, default is 100
seed	random seed, default is 31415
force9	Optional. There are cases where DK data doesn't have DK. But we need the entire matrix. By default it is FALSE.

Value

list with standard error of parameters, estimates of learning, standard error of learning by item

Examples

```
pre_test <- data.frame(pre_item1 = c(1,0,0,1,0), pre_item2 = c(1,NA,0,1,0))
pst_test <- data.frame(pst_item1 = pre_test[,1] + c(0,1,1,0,0),
             pst_item2 = pre_test[,2] + c(0,1,0,0,1))
## Not run: lca_se(pre_test, pst_test, nsamps = 10, seed = 31415)
```

---

multi_transmat *Creates a transition matrix for each item.*

Description

Needs an 'interleaved' dataframe (see interleave function). Pre-test item should be followed by corresponding post-item item etc. Don't knows must be coded as NA. Function handles items without don't know responses. The function is used internally. It calls transmat.

Usage

```
multi_transmat(
  pre_test = NULL,
  pst_test = NULL,
  subgroup = NULL,
  force9 = FALSE,
  agg = FALSE
)
```

Arguments

pre_test	Required. data.frame carrying responses to pre-test questions.
pst_test	Required. data.frame carrying responses to post-test questions.
subgroup	a Boolean vector identifying the subset. Default is NULL.
force9	Optional. There are cases where DK data doesn't have DK. But we need the entire matrix. By default it is FALSE.
agg	Optional. Boolean. Whether or not to add a row of aggregate transitions at the end of the matrix. Default is FALSE.

Details

multi_transmat: transition matrix of all the items

Value

matrix with rows = total number of items + 1 (last row contains aggregate distribution across items) number of columns = 4 when no don't know, and 9 when there is a don't know option

Examples

```
pre_test <- data.frame(pre_item1 = c(1,0,0,1,0), pre_item2 = c(1,NA,0,1,0))
pst_test <- data.frame(pst_item1 = pre_test[,1] + c(0,1,1,0,0),
             pst_item2 = pre_test[,2] + c(0,1,0,0,1))
multi_transmat(pre_test, pst_test)
```

---

nona *No NAs*

Description

Converts NAs to 0s

Usage

```
nona(vec = NULL)
```

Arguments

vec	Required. Character or Numeric vector.

Value

Character vector.

Examples

```
x <- c(NA, 1, 0); nona(x)
x <- c(NA, "dk", 0); nona(x)
```

stnd_cor *Standard Guessing Correction for Learning*

Description

Estimate of learning adjusted with standard correction for guessing. Correction is based on number of options per question. The function takes separate pre-test and post-test dataframes. Why do we need dataframes? To accomodate multiple items. The items can carry NA (missing). Items must be in the same order in each dataframe. Assumes that respondents are posed same questions twice. The function also takes a `lucky` vector — the chance of getting a correct answer if guessing randomly. Each entry is 1/(number of options). The function also optionally takes a vector carrying names of the items. By default, the vector carrying adjusted learning estimates takes same item names as the pre_test items. However you can assign a vector of names separately via `item_names`.

Usage

```
stnd_cor(pre_test = NULL, pst_test = NULL, lucky = NULL, item_names = NULL)
```

Arguments

`pre_test`	Required. data.frame carrying responses to pre-test questions.
`pst_test`	Required. data.frame carrying responses to post-test questions.
`lucky`	Required. A vector. Each entry is 1/(number of options)
`item_names`	Optional. A vector carrying item names.

Value

a list of three vectors, carrying pre-treatment corrected scores, post-treatment scores, and adjusted estimates of learning

Examples

```
# Without DK
pre_test <- data.frame(item1 = c(1,0,0,1,0), item2 = c(1,NA,0,1,0))
pst_test <- pre_test + cbind(c(0,1,1,0,0), c(0,1,0,0,1))
lucky <- rep(.25, 2); stnd_cor(pre_test, pst_test, lucky)
# With DK
pre_test <- data.frame(item1 = c(1,0,0,1,0,'d',0), item2 = c(1,NA,0,1,0,'d','d'))
pst_test <- data.frame(item1 = c(1,0,0,1,0,'d',1), item2 = c(1,NA,0,1,0,1,'d'))
lucky <- rep(.25, 2); stnd_cor(pre_test, pst_test, lucky)
```

transmat *transmat: Cross-wave transition matrix*

Description

Prints Cross-wave transition matrix and returns the vector behind the matrix. Missing values are treated as ignorance. Don't know responses need to be coded as 'd'.

Usage

```
transmat(pre_test_var, pst_test_var, subgroup = NULL, force9 = FALSE)
```

Arguments

pre_test_var	Required. A vector carrying pre-test scores of a particular item. Only
pst_test_var	Required. A vector carrying post-test scores of a particular item
subgroup	Optional. A Boolean vector indicating rows of the relevant subset.
force9	Optional. There are cases where DK data doesn't have DK. But we need the entire matrix. By default it is FALSE.

Value

a numeric vector. Assume 1 denotes correct answer, 0 and NA incorrect, and d 'don't know.' When there is no don't know option and no missing, the entries are: x00, x10, x01, x11 When there is a don't know option, the entries of the vector are: x00, x10, xd0, x01, x11, xd1, xd0, x1d, xdd

Examples

```
pre_test_var <- c(1,0,0,1,0,1,0)
pst_test_var <- c(1,0,1,1,0,1,1)
transmat(pre_test_var, pst_test_var)

# With NAs
pre_test_var <- c(1,0,0,1,"d","d",0,1,NA)
pst_test_var <- c(1,NA,1,"d",1,0,1,1,"d")
transmat(pre_test_var, pst_test_var)
```

validate_compatible_dataframes *Validate that two data frames have compatible dimensions*

Description

Validate that two data frames have compatible dimensions

Usage

```
validate_compatible_dataframes(pre_test, pst_test)
```

Arguments

pre_test	pre-test data frame
pst_test	post-test data frame

Value

TRUE if valid, throws error otherwise

---

validate_gamma *Validate gamma parameter*

Description

Validate gamma parameter

Usage

```
validate_gamma(gamma)
```

Arguments

gamma	probability parameter

Value

TRUE if valid, throws error otherwise

---

validate_lucky_vector *Validate lucky vector for standard correction*

Description

Validate lucky vector for standard correction

Usage

```
validate_lucky_vector(lucky, n_items)
```

Arguments

lucky	vector of guessing probabilities
n_items	number of items to validate against

Value

TRUE if valid, throws error otherwise

`validate_priors` *Validate prior parameters*

Description

Validate prior parameters

Usage

```
validate_priors(priors, expected_length, param_name)
```

Arguments

`priors`	vector of prior parameters
`expected_length`	expected length of priors vector
`param_name`	name of parameter for error messages

Value

TRUE if valid, throws error otherwise

`validate_transition_values` *Validate transition matrix values*

Description

Validate transition matrix values

Usage

```
validate_transition_values(pre_test_var, pst_test_var)
```

Arguments

`pre_test_var`	pre-test variable vector
`pst_test_var`	post-test variable vector

Value

TRUE if valid, throws error otherwise

Index