Package: guess (via r-universe)

September 3, 2024

Title Adjust Estimates of Learning for Guessing

Version 0.2.0

Maintainer Gaurav Sood <gsood07@gmail.com>

Description Adjust Estimates of Learning for Guessing. The package provides standard guessing correction, and a latent class model that leverages informative pre-post transitions. For details of the latent class model, see <http://gsood.com/research/papers/guess.pdf>.

URL http://github.com/soodoku/guess

BugReports http://github.com/soodoku/guess/issues

Depends R (>= 3.2.1)

Imports Rsolnp

License MIT + file LICENSE

LazyData true

VignetteBuilder knitr

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

RoxygenNote 6.0.1

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

RemoteUrl https://github.com/soodoku/guess

RemoteRef HEAD

RemoteSha 47b2b1b9e263f49651fe6b3aa09ed50ffca685ea

Contents

`alldat` *Simulated Responses to Knowledge Questions Without Don't Know*

Description

Simulate Pre- and Post-Process Responses to Knowledge Questions without Don't Know

Simulate Pre- and Post-Process Responses to Knowledge Questions with Don't Know

Usage

```
alldat

alldat_dk
```

Format

A data frame with 500 rows and 4000 variables:

`dk_sim` *Simulated Responses to Knowledge Questions With Don't Know*

Description

Simulate Pre- and Post-Process Responses to Knowledge Questions with Don't Know

Usage

```
dk_sim
```

Format

A data frame with 500 rows and 4000 variables:

`dk_sim_params` *Simulated Responses to Knowledge Questions With Don't Know*

Description

Simulate Pre- and Post-Process Responses to Knowledge Questions with Don't Know

Usage

```
dk_sim_params
```

Format

A data frame with 1000 rows and 5 variables:

`fit_dk` *Goodness of fit statistics for data with don't know*

Description

For data with Don't Know, chi-square goodness of fit between true and model based multivariate distribution

Usage

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

Arguments

`pre_test`	data.frame carrying pre_test items
`pst_test`	data.frame carrying pst_test items
`g`	estimates of γ produced from `lca_cor`
`est.param`	estimated parameters produced from `lca_cor`
`force9`	Optional. There are cases where DK data doesn't have DK. But we need the entire matrix. By default it is FALSE.

Details

fit_nodk

Value

matrix with two rows: top row carrying chi-square value, and bottom row probability of observing that value

Examples

```
## Not run: fit_dk(pre_test, pst_test, g, est.param)
```

---

fit_nodk *Goodness of fit statistics for data without don't know*

Description

For data without Don't Know, chi-square goodness of fit between true and model based multivariate distribution

Usage

```
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 γ produced from `lca_cor`
`est.param`	estimated parameters produced from `lca_cor`

Details

fit_nodk

Value

matrix with two rows: top row carrying chi-square value, and bottom row probability of observing that value

Examples

```
## Not run: fit_nodk(pre_test, pst_test, g, est.param)
```

`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)
```

`guess` **guess** *adjust estimates of learning for guessing related bias.*

Description

It implements the method discussed in http://gsood.com/research/papers/guess.pdf

`interleave` *Interleave*

Description

Used Internally Interleave two character vectors. The output is a vector. The first entry is from the first vector. Vectors can be of different lengths. If one is shorter than the other, entries of unmatched longer vector are left un-interleaved.

Usage

```
interleave(vec1, vec2)
```

Arguments

`vec1`	first vector
`vec2`	second vector

Value

interleaved vector # t1 <- paste0("t1", letters[1:5]); t2 <- paste0("t2", letters[1:5]); interleave(t1, t2)

`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)
```

`params` *Parameters of Simulated Responses to Knowledge Questions Without Don't Know*

Description

Paramaters of the simulated Pre- and Post-Process Responses to Knowledge Questions without Don't Know

Usage

```
params
```

Format

A data frame with 1000 rows and 4 variables:

`params_dk` *Parameters of Simulated Responses to Knowledge Questions With Don't Know*

Description

Paramaters of the simulated Pre- and Post-Process Responses to Knowledge Questions with Don't Know

Usage

```
params_dk
```

Format

A data frame with 1000 rows and 4 variables:

`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)
```

Index