Package: examly (via r-universe)

May 15, 2026

Title Statistical Metrics and Reporting Tool

Version 0.3

Description A 'Shiny'-based toolkit for item/test analysis. It is designed for multiple-choice, true-false, and open-ended questions. The toolkit is usable with datasets in 1-0 or other formats. Key analyses include difficulty, discrimination, response-option analysis, reports. The classical test theory methods used are described in Ebel & Frisbie (1991, ISBN:978-0132892314).

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1)

Imports shiny, dplyr, ggplot2, tidyr, purrr, stringr, readr, readxl, officer, flextable, glue, magrittr, jsonlite, tibble, htmltools

Suggests testthat (>= 3.0.0), knitr, rmarkdown, spelling, psychometric

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/ahmetcaliskan1987/examly

BugReports https://github.com/ahmetcaliskan1987/examly/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Language en-US

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

Repository https://ahmetcaliskan1987.r-universe.dev

Date/Publication 2026-05-15 06:56:45 UTC

RemoteUrl https://github.com/ahmetcaliskan1987/examly

RemoteRef HEAD

RemoteSha ee3ca510f4e714ab3833071bad9d63ddb15983c4

Contents

color_badge . . . 2
comment_overall_keys . . . 3
d_mode . . . 3
detect_id_cols . . . 4
difficulty_label_key . . . 4
discrimination_decision_key . . . 5
get_itemexam_quant . . . 6
i18n_load . . . 6
i18n_t . . . 7
is_scored_01 . . . 8
kr20 . . . 8
norm_cols . . . 9
norm_letter . . . 9
parse_lc_bin . . . 10
parse_lc_raw . . . 10
parse_mc_bin . . . 11
parse_tf_bin . . . 11
pbiserial_rest . . . 12
q_index . . . 12
run_app . . . 13
student_counts . . . 14

Index **15**

`color_badge` *Create a colored HTML span badge (UI helper)*

Description

Generates a colored HTML badge for displaying values (like p-values or correlations) in the Shiny UI.

Usage

```
color_badge(v, type = c("generic", "p", "r"))
```

Arguments

`v`	The numeric value to display.
`type`	The type of value ('generic', 'p', 'r') for coloring rules.

Value

An `htmltools::span()` object.

Examples

```
if (interactive()) {
  # Example for difficulty (p)
  color_badge(0.5, "p")
  # Example for discrimination (r)
  color_badge(0.2, "r")
}
```

---

comment_overall_keys *Generate translation keys for overall comments*

Description

Creates a vector of translation keys based on average difficulty (ap) and discrimination (ar).

Usage

```
comment_overall_keys(ap, ar)
```

Arguments

ap	Average difficulty (p-value).
ar	Average discrimination (r-value).

Value

A character vector of translation keys.

Examples

```
comment_overall_keys(0.6, 0.35) # Medium, Keep
comment_overall_keys(0.2, 0.15) # Hard, Remove
```

---

d_mode *Calculate the mode*

Description

Finds the most frequent value (the mode) in a vector.

Usage

```
d_mode(x)
```

Arguments

x	A vector.

Value

The mode of the vector. Returns NA if the vector is empty.

Examples

```
d_mode(c(1, 2, 2, 3, 3, 3, 4))
d_mode(c("a", "b", "a"))
```

---

detect_id_cols *Detect ID columns using regex*

Description

Searches a vector of column names for common ID-related patterns.

Usage

```
detect_id_cols(cols)
```

Arguments

cols	A character vector of column names.

Value

A character vector of names that matched the ID pattern.

Examples

```
detect_id_cols(c("Ad", "Soyad", "ogrenci no", "Madde1", "StudentID"))
```

---

difficulty_label_key *Generate label key for difficulty (p)*

Description

Returns a specific translation key based on an item's difficulty value.

Usage

```
difficulty_label_key(p)
```

Arguments

p	A numeric item difficulty value.

Value

A character string (translation key).

Examples

```
difficulty_label_key(0.3) # Hard
difficulty_label_key(0.7) # Medium
difficulty_label_key(0.9) # Easy
```

`discrimination_decision_key` *Generate label key for discrimination (r)*

Description

Returns a specific translation key based on an item's discrimination value.

Usage

```
discrimination_decision_key(r)
```

Arguments

`r`	A numeric item discrimination value.

Value

A character string (translation key).

Examples

```
discrimination_decision_key(0.15) # Remove
discrimination_decision_key(0.25) # Consider
discrimination_decision_key(0.4)  # Keep
```

get_itemexam_quant *Safely get quantile from 'psychometric' package*

Description

Tries to get the default `quant` argument from `psychometric::item.exam`. Returns 0.27 if the package is not installed or an error occurs.

Usage

```
get_itemexam_quant()
```

Value

A numeric value for the quantile (default 0.27).

Examples

```
get_itemexam_quant()
```

i18n_load *Load translation dictionary*

Description

Finds `<lang>.json` under the installed package's `i18n/` folder and, if not found (during development), falls back to `inst/i18n/<lang>.json`.

Usage

```
i18n_load(lang = "tr")
```

Arguments

`lang`	Character scalar language code. Currently `"tr"` or `"en"`.

Value

A named list (key -> string) parsed from the JSON file.

Examples

```
# Always-fast: locate the installed TR dictionary (empty string if not installed)
system.file("shinyapp", "i18n", "tr.json", package = "examly")

# Safe example that runs only if the file actually exists
p <- system.file("shinyapp", "i18n", "tr.json", package = "examly")
if (nzchar(p) && file.exists(p)) {
  d <- i18n_load("tr")
  i18n_t(d, "ui.title", "Baslik")
}
```

`i18n_t` *Translate a UI/message key*

Description

Returns the value for `key` from a dictionary produced by `i18n_load()`. If the key is missing, returns `default` when provided, otherwise the key itself.

Usage

```
i18n_t(dict, key, default = NULL)
```

Arguments

`dict`	Named list produced by `i18n_load()`.
`key`	Character scalar; lookup key.
`default`	Optional fallback value if the key is not present.

Value

Character scalar.

Examples

```
# A quick, fully automatic check:
p <- system.file("shinyapp", "i18n", "en.json", package = "examly")
if (nzchar(p) && file.exists(p)) {
  d <- i18n_load("en")
  i18n_t(d, "buttons.download", "Download")
}
```

is_scored_01 *Check if a vector is scored 0/1*

Description

Detects if a vector (after removing NAs) contains only 0 and 1.

Usage

```
is_scored_01(vec)
```

Arguments

vec	The vector to check.

Value

TRUE if the vector is 0/1 scored, FALSE otherwise.

Examples

```
is_scored_01(c(1, 0, 1, 0, NA))
is_scored_01(c(1, 0, 2, 0))
is_scored_01(c("A", "B", "C"))
```

kr20 *Calculate KR-20 reliability coefficient*

Description

Calculates the Kuder-Richardson 20 (KR-20) reliability coefficient for a data.frame or matrix of dichotomous (0/1) items.

Usage

```
kr20(m)
```

Arguments

m	A data.frame or matrix where rows are subjects and columns are dichotomously scored (0/1) items.

Value

A numeric value for the KR-20 coefficient, or NA_real_ if calculation is not possible.

Examples

```
item_matrix <- data.frame(
  m1 = c(1, 1, 0, 1),
  m2 = c(1, 0, 1, 1),
  m3 = c(0, 1, 0, 0)
)
kr20(item_matrix)
```

`norm_cols` *Normalize Column Names*

Description

A helper function that takes a mixed character vector or a single semi-colon/comma-separated string and returns a clean character vector of column names.

Usage

```
norm_cols(x)
```

Arguments

`x`	A character vector or a single string containing column names.

Value

A character vector of trimmed, non-empty column names.

Examples

```
norm_cols(" m1 , m2;m3")
norm_cols(c(" m1 ", "m2", "", " m3 "))
```

`norm_letter` *Normalize letter grades*

Description

Cleans and validates a vector of characters, keeping only standard letter grades (A, B, C, D, E).

Usage

```
norm_letter(x)
```

Arguments

`x`	A vector, typically character.

Value

A character vector of normalized grades (A-E) or NA.

Examples

```
norm_letter(c(" a ", "B", "c", "F", "d", NA))
```

`parse_lc_bin` *Parse 1/0 coded data*

Description

Validates that a vector contains only 1s, 0s, or NAs.

Usage

```
parse_lc_bin(x)
```

Arguments

`x`	A vector of potential 1/0 scores.

Value

An integer vector (1, 0, or NA).

Examples

```
parse_lc_bin(c(1, 0, "1", "0", 2, "A", NA))
```

`parse_lc_raw` *Parse raw continuous/Likert scores*

Description

Cleans a vector of potential scores, converting to numeric and removing invalid or out-of-range values.

Usage

```
parse_lc_raw(x)
```

Arguments

`x`	A vector (typically character) of raw scores.

Value

A numeric vector of cleaned scores.

Examples

```
parse_lc_raw(c("10", "5.5", "0", "-2", "ikiyüz", NA, "1000001"))
```

---

parse_mc_bin *Score Multiple Choice items as 1/0*

Description

Scores a Multiple Choice (A-E) response vector against a key.

Usage

```
parse_mc_bin(x, key)
```

Arguments

x	A vector of student responses.
key	The correct answer key (e.g., "A", "B").

Value

An integer vector (1=correct, 0=wrong, NA=invalid).

Examples

```
parse_mc_bin(c("a", "B", "c", "F", " b "), key = "B")
```

---

parse_tf_bin *Score True/False items as 1/0*

Description

Scores a True/False (Doğru/Yanlış) response vector against a key.

Usage

```
parse_tf_bin(x, key)
```

Arguments

x	A vector of student responses.
key	The correct answer key (e.g., "D", "Y", "TRUE", "FALSE").

Value

An integer vector (1=correct, 0=wrong, NA=invalid).

Examples

```
parse_tf_bin(c("D", "Y", "DOGRU", "False", "Belki"), key = "D")
```

`pbiserial_rest` *Point-biserial correlation for item analysis*

Description

Calculates the correlation between a single item's score and the rest of the total score.

Usage

```
pbiserial_rest(item, rest)
```

Arguments

`item`	A numeric vector of dichotomous item scores (0/1).
`rest`	A numeric vector of the total scores, excluding the item.

Value

The point-biserial correlation coefficient.

Examples

```
item1 <- c(1, 0, 1, 0, 1, 1)
rest_score <- c(10, 8, 12, 5, 9, 11)
pbiserial_rest(item1, rest_score)
```

`q_index` *Calculate q-index (1 - p)*

Description

A simple helper to calculate the inverse of the p-value (difficulty index).

Usage

```
q_index(p)
```

Arguments

`p`	A numeric value or vector (item difficulty).

Value

A numeric value or vector (1 - p).

Examples

```
q_index(0.8)
q_index(c(0.2, 0.5, 0.7))
```

`run_app` *Launch examly Shiny application*

Description

Launches the packaged Shiny app located in `inst/shinyapp`. If the app files are not found, a minimal placeholder app is launched instead.

Usage

```
run_app()
```

Details

This function is exported so users can run `examly::run_app()`.

Value

Invisibly returns `NULL`. Called for its side effects.

Examples

```
system.file("shinyapp", package = "examly")
if(interactive()){
examly::run_app()
}
```

`student_counts` *Student-level counts (Correct/Incorrect/Missing)*

Description

Calculates the total number of correct, incorrect, and missing answers for each student (row).

Usage

```
student_counts(sc)
```

Arguments

`sc`	A data.frame of scored items (0=wrong, 1=correct, NA=missing).

Value

A tibble with columns `Dogru`, `Yanlis`, and `Bos`.

Examples

```
score_df <- data.frame(m1 = c(1, 0, 1), m2 = c(0, 1, NA), m3 = c(1, 1, 1))
student_counts(score_df)
```

Index

color_badge, 2
comment_overall_keys, 3

d_mode, 3
detect_id_cols, 4
difficulty_label_key, 4
discrimination_decision_key, 5

get_itemexam_quant, 6

i18n_load, 6
i18n_load(), 7
i18n_t, 7
is_scored_01, 8

kr20, 8

norm_cols, 9
norm_letter, 9

parse_lc_bin, 10
parse_lc_raw, 10
parse_mc_bin, 11
parse_tf_bin, 11
pbiserial_rest, 12

q_index, 12

run_app, 13

student_counts, 14