Package: ResponseScan (via r-universe)

November 8, 2026

Type Package

Title Quick Screening of Response Patterns in Test and Survey Items

Version 0.4.1

Description One call screens the item responses of a test or questionnaire for patterns that deserve a closer look before modelling: flat or long-string responding, low intra-individual variability, multivariate outliers, incoherent psychometric synonyms or even-odd halves, overuse of the midpoint or of the extremes, duplicated response vectors and identical patterns that arrive in series of consecutive submissions. Indices follow Curran (2016) <doi:10.1016/j.jesp.2015.07.006>; person fit in item response theory (the standardized log-likelihood) is optional, and cut-offs can be taken from a reference group. Results are returned per person, with flags, frequency tables of response patterns and plots. Whole batches of data (a second collection phase, one classroom) can be compared with a reference group: the item correlation structure, with permutation tests that leave out flat responders and match the response level, and the Jennrich (1970) <doi:10.1080/01621459.1970.10481133> test; the similarity of consecutive rows; and near-duplicate respondents. Flagged respondents can be removed in one step and the same confirmatory or exploratory factor model can be refitted with and without them.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1)

Imports ggplot2 (>= 3.4.0), stats, utils

Suggests lavaan, mirt, testthat (>= 3.0.0)

LazyData true

Config/testthat/edition 3

NeedsCompilation no

Author Jose Ventura-Leon [aut, cre] (ORCID:
<https://orcid.org/0000-0003-2996-4244>)

Maintainer Jose Ventura-Leon <jventuraleon@gmail.com>

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

Date/Publication 2026-11-08 19:50:40 UTC

RemoteUrl https://github.com/cran/ResponseScan

RemoteRef HEAD

RemoteSha 2fb4cb6b8d3ce217afd195326b5cce0d1eb57777

Contents

batch_comparison-methods 2
batch_sim 3
compare_batches 4
compare_fit 7
exclude_responses 8
fit_comparison-methods 10
neighbor_test 11
neighbor_test-methods 13
order_test 14
order_test-methods 15
plot.response_scan 16
response_scan-methods 18
scan_responses 19
survey_sim 22

Index **24**

`batch_comparison-methods`

Print and plot a batch comparison

Description

`print()` shows the groups compared and the tests. `plot()` draws one of three views, with **ggplot2**:

`"null"` Histograms of each statistic in the reference samples, with the focal value as a vertical line. A line far to the left of the histogram means less structure than any comparable group of reference respondents. Compare the panels of the full and the level-matched reference: when only the first one is extreme, the level explains the difference.

`"heatmap"` Item correlations of the reference (level-matched when available) and of the focal group, side by side, with the scales marked. Genuine data show blocks along the diagonal.

`"structure"` Mean correlation within and between scales in each group. The gap between both bars is the structure.

Usage

```
## S3 method for class 'batch_comparison'
print(x, digits = 3, ...)

## S3 method for class 'batch_comparison'
plot(
  x,
  type = c("null", "heatmap", "structure"),
  lang = c("en", "es"),
  base_size = 11,
  ...
)
```

Arguments

x	An object returned by compare_batches().
digits	Number of decimals printed.
...	Not used.
type	One of "null", "heatmap" or "structure".
lang	Language of the labels: "en" or "es".
base_size	Base font size of the theme.

Value

print() returns x invisibly. plot() returns a ggplot object, printed on the current device and returned invisibly.

Examples

```
sc <- list(F1 = paste0("I", 1:4), F2 = paste0("I", 5:8), F3 = paste0("I", 9:12))
cb <- compare_batches(batch_sim, group = "phase", scales = sc, B = 200)
print(cb, digits = 2)
plot(cb, type = "structure")
```

---

batch_sim *Simulated two-phase survey with a hand-filled block*

Description

Simulated answers of 260 respondents to a 12-item questionnaire (three scales of four items, options 1 to 5), collected in two phases. Phase 1 has 200 genuine respondents, 8 of them flat. Phase 2 has 30 genuine respondents with a higher level and a block of 30 rows filled by hand: each row copies the previous one and changes two or three answers, and the sex and age are often copied too. The block has the level of phase 2 but no factor structure. It is the example data of compare_batches(), order_test() and neighbor_test(). No real person is included.

Usage

```
batch_sim
```

Format

A data frame with 260 rows and 17 columns:

id Respondent number, in order of arrival.

phase Collection phase, 1 or 2.

sex `"Female"` or `"Male"`.

age Age group: `"17-20"`, `"21-25"`, `"26-30"` or `"31+"`.

source How the row was generated: `"genuine"`, `"flat"` or `"hand-filled"`.

I1 Item 1 (scale 1); options 1 to 5.

I2 Item 2 (scale 1); options 1 to 5.

I3 Item 3 (scale 1); options 1 to 5.

I4 Item 4 (scale 1); options 1 to 5.

I5 Item 5 (scale 2); options 1 to 5.

I6 Item 6 (scale 2); options 1 to 5.

I7 Item 7 (scale 2); options 1 to 5.

I8 Item 8 (scale 2); options 1 to 5.

I9 Item 9 (scale 3); options 1 to 5.

I10 Item 10 (scale 3); options 1 to 5.

I11 Item 11 (scale 3); options 1 to 5.

I12 Item 12 (scale 3); options 1 to 5.

Source

Simulated with the script `data-raw/make_batch_sim.R` of the package source.

`compare_batches` *Compare the correlation structure of a batch of data with a reference*

Description

Asks whether a batch of respondents (a second collection phase, one classroom, the cases entered by one person) answers with the same structure as the rest. In genuine data, items of the same scale move together more than items of different scales, because the answers come from the respondents' traits. Answers filled by hand can copy the level (many 4s and 5s) but rarely reproduce that structure.

Usage

```
compare_batches(
  data,
  group,
  items = NULL,
  scales = NULL,
  focal = NULL,
  reference = NULL,
  drop_flat = TRUE,
  match_level = TRUE,
  B = 2000,
  min_ratio = 3
)
```

Arguments

`data`	A data frame with one row per respondent.
`group`	The batch of each respondent: a column name of `data` or a vector of length `nrow(data)`.
`items`	Item columns. By default, the items in `scales` or every numeric column except `group`.
`scales`	Optional named list of character vectors, one per scale. Needed for the `structure` statistic.
`focal`	Group (or groups) under examination. By default, the smallest group.
`reference`	Group (or groups) used as reference. By default, every other group.
`drop_flat`	Logical; remove respondents with the same answer in every item before comparing.
`match_level`	Logical; add the comparison with reference samples paired by level with the focal batch.
`B`	Number of random samples drawn from the reference.
`min_ratio`	Minimum ratio between the reference and the focal group for the whole batch to be compared in every sample. With a smaller reference (even one smaller than the batch) the batch is subsampled to `floor(n_reference / min_ratio)` respondents. Larger values make the control vary more but leave smaller focal samples, with less power. Use `1` for the behaviour of version 0.4.0 when the reference is at least as large as the batch.

Details

Two statistics are computed in the focal batch: the mean inter-item correlation and, when `scales` defines two or more scales, the `structure` (mean correlation within scales minus mean correlation between scales). Their null distribution comes from `B` random samples, of the size of the focal batch, drawn from the reference group. The p value is the share of samples with a value as low or lower than the focal one, so small values mean that the batch has less structure than any comparable group of reference respondents.

Two precautions avoid false alarms, and both changed the conclusion in the real case that motivated this function:

- `drop_flat`: respondents with the same answer in every item inflate the correlations of the group they belong to. They are removed from both groups before comparing.
- `match_level`: a batch that answers high has less variability, and less variability lowers correlations on its own (range restriction). Each reference sample is then built by pairing every focal respondent with the reference respondent of the closest mean answer, so it reproduces the whole distribution of levels of the batch. When the difference disappears with this control, the low correlations are explained by the level, not by the answers.

The level-matched samples need a reference clearly larger than the batch, above all when the batch answers at an unusual level. When the reference has fewer than `min_ratio` times the focal respondents, pairing every focal respondent picks nearly the same reference rows in every sample, the control does not vary and its p values overstate the evidence. In the real case behind this argument (32 items, a reference of 150 and a lower-level batch of 210), consecutive matched samples shared 100% of their rows with the whole reference, 87% with `min_ratio = 2` and 63% with `min_ratio = 3`. In that case each sample takes a random subsample of the batch, of the size of the reference divided by `min_ratio`, and pairs the reference with it; the focal value reported is then the median over the focal subsamples, so that both sides are computed at the same size. This median is more stable than a single focal sample, which makes the test somewhat conservative. A warning is given when consecutive level-matched samples still share more than 80% of their rows.

The Jennrich (1970) test of the equality of the two correlation matrices is reported as well. It is very sensitive: it also rejects when two genuine populations differ, so it says that the matrices differ, not why.

Value

An object of class `batch_comparison`, a list with:

groups One row per group (focal, reference and, if requested, the level-matched reference): respondents, flat respondents removed, respondents used, mean answer, mean within-person standard deviation, mean inter-item correlation and the within, between and structure correlations.

tests One row per comparison and statistic: the focal value, the size of the samples compared (`n_sample`), the median and the 2.5th percentile of the reference samples, and the p value.

jennrich Jennrich test of the focal against each reference: chi- square, degrees of freedom and p value.

null, cor, settings The values of the reference samples, the correlation matrices and the arguments used, kept for the plots. `settings` also keeps the focal sample size (`focal_sample`), whether the batch was subsampled and the share of rows shared by consecutive level-matched samples (`matched_overlap`).

References

Jennrich, R. I. (1970). An asymptotic chi-square test for the equality of two correlation matrices. *Journal of the American Statistical Association, 65*(330), 904-912. doi:10.1080/01621459.1970.10481133

See Also

order_test() and neighbor_test() for traces of copying, and the reference argument of scan_responses() to flag single respondents with cut-offs taken from a reference group.

Examples

```
sc <- list(F1 = paste0("I", 1:4), F2 = paste0("I", 5:8), F3 = paste0("I", 9:12))
cb <- compare_batches(batch_sim, group = "phase", scales = sc, B = 200)
cb
plot(cb)
plot(cb, type = "heatmap")
plot(cb, type = "structure", lang = "es")
```

---

compare_fit *Fit the same CFA or EFA with and without flagged respondents*

Description

A sensitivity analysis in one call: the same factor model is fitted to the full sample and to the data left after each exclusion, and the fit indices, the mean standardized loading and the factor correlations are put side by side. Models are fitted with **lavaan**, which must be installed.

Usage

```
compare_fit(
  x,
  data,
  model = NULL,
  exclusions = NULL,
  type = c("cfa", "efa"),
  nfactors = 1,
  items = NULL,
  ordered = TRUE,
  ...
)
```

Arguments

x	An object returned by scan_responses().
data	The data frame that was screened (same rows, same order).
model	For type = "cfa", a model in lavaan syntax. Ignored for type = "efa".
exclusions	Named list of exclusions. Each element is NULL (full sample), a character vector of flags (see exclude_responses(); "any" is allowed), a logical vector of rows to remove, or a list with any of the arguments flags, patterns, min_flags and rows of exclude_responses(). By default: the full sample, each flag that marks someone, and any flag.

type	"cfa" or "efa".
nfactors	Number of factors for type = "efa".
items	Items analysed by the EFA. By default, every screened item.
ordered	Logical; treat the items as ordinal (the default, with the WLSMV estimator) or as continuous.
...	Further arguments passed to lavaan::cfa() or lavaan::efa() (for example estimator or rotation).

Value

An object of class fit_comparison, a list with table (one row per exclusion: removed and kept respondents, chi-square, degrees of freedom, CFI, TLI, RMSEA, SRMR, mean_loading (mean over items of the largest absolute standardized loading of each item), the factor correlations and a note with warnings or errors; the scaled versions of the indices are used when the estimator provides them), fits (the lavaan objects, NULL when a model failed) and data (the data set used in each fit). It has print() and plot() methods.

Examples

```
if (requireNamespace("lavaan", quietly = TRUE)) {
  res <- scan_responses(survey_sim,
    scales = list(A = paste0("A", 1:8), B = paste0("B", 1:10)),
    reverse = paste0("A", 1:4), time = "submitted",
    pattern_items = paste0("A", 1:8))
  m <- "A =~ A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8
        B =~ B1 + B2 + B3 + B4 + B5 + B6 + B7 + B8 + B9 + B10"
  cf <- compare_fit(res, survey_sim, model = m,
    exclusions = list("Full sample" = NULL, "Without series" = "serial",
                      "Without any flag" = "any"))
  cf
  plot(cf)
}
```

---

exclude_responses *Remove flagged respondents or response patterns from the data*

Description

Returns the original data without the respondents that meet the chosen criteria, ready to be passed to any model (for example lavaan::cfa(), lavaan::efa() or psych::fa()). Criteria can be combined: a respondent is removed when any of them applies.

Usage

```
exclude_responses(
  x,
  data,
  flags = "serial",
  patterns = NULL,
  min_flags = NULL,
  rows = NULL
)
```

Arguments

x	An object returned by scan_responses().
data	The data frame that was screened (same rows, same order). All its columns are kept.
flags	Character vector with flag names ("flat", "low_irv", "mahal", "psychsyn", "duplicate", "serial", "zh") or "any" for respondents with at least one flag. NULL uses no flag.
patterns	Character vector with response patterns, as they appear in x$patterns$pattern; every respondent with one of these patterns is removed, whether or not it arrived in series.
min_flags	Remove respondents with at least this number of flags.
rows	Additional row positions (integer) or a logical vector of length nrow(data) to remove.

Value

A data frame with the kept rows of data (original row names are preserved) and two attributes: excluded, the positions of the removed rows, and rule, a text that describes the criteria used. A warning is given when the rule removes more than half of the sample.

Examples

```
res <- scan_responses(survey_sim, items = paste0("A", 1:8),
                      reverse = paste0("A", 1:4), time = "submitted")
clean <- exclude_responses(res, survey_sim, flags = "serial")
nrow(clean)
attr(clean, "rule")
# The same pattern, also when it did not arrive in series
nrow(exclude_responses(res, survey_sim, flags = NULL, patterns = "11113333"))
# Respondents with two or more flags
nrow(exclude_responses(res, survey_sim, flags = NULL, min_flags = 2))
```

`fit_comparison-methods` *Print and plot a fit comparison*

Description

Print and plot a fit comparison

Usage

```
## S3 method for class 'fit_comparison'
print(x, digits = 3, ...)

## S3 method for class 'fit_comparison'
plot(x, lang = c("en", "es"), base_size = 11, ...)
```

Arguments

`x`	An object returned by `compare_fit()`.
`digits`	Number of decimals.
`...`	Not used.
`lang`	Language of the labels: `"en"` or `"es"`.
`base_size`	Base font size of the **ggplot2** theme.

Details

The plot has one panel per fit index, with the usual reference values as dashed lines (CFI and TLI .95, RMSEA .06, SRMR .08), and one panel for the factor correlations, if any, with a dashed line at 1. Each row is one exclusion. If the points barely move from row to row, the decision to exclude does not change the conclusions; if a point crosses a reference line, or the factor correlation changes a lot, the decision matters and should be reported. Exclusions with an improper solution are drawn in red.

Value

`print()` returns `x` invisibly. `plot()` returns a `ggplot` object, printed on the current device and returned invisibly.

Examples

```
if (requireNamespace("lavaan", quietly = TRUE)) {
  res <- scan_responses(survey_sim, items = paste0("B", 1:10))
  cf <- compare_fit(res, survey_sim, type = "efa", nfactors = 1)
  print(cf, digits = 2)
  plot(cf, lang = "es")
}
```

neighbor_test *Test for near-duplicate respondents*

Description

Exact duplicates are easy to spot; edited copies (a row copied and then changed in two or three items) are not. For every respondent this function counts the items in which they differ from their most similar respondent (the nearest neighbour) and compares the median with two baselines:

- Shuffled data: each item is shuffled on its own, which keeps the distribution of every item but breaks any link between rows. Genuine respondents are always somewhat closer than shuffled ones, because people with a high trait answer high almost everywhere, so this baseline alone is not evidence of copying.
- A control (when `group` is given): samples of the reference group, of the size of the focal batch and of the same level when `match_level = TRUE`, go through the same comparison. The statistic is the gap between shuffled and real distances. A focal gap well above the control gaps means rows closer to each other than genuine data of the same level would be.

As in `compare_batches()`, the control needs a reference clearly larger than the batch. When the reference has fewer than `min_ratio` times the focal respondents, each control sample is paired with a random subsample of the batch of the same size, and `p_control` compares the median gap of those focal subsamples (`focal_gap_sample`) with the control gaps.

Usage

```
neighbor_test(
  data,
  items = NULL,
  group = NULL,
  focal = NULL,
  reference = NULL,
  drop_flat = TRUE,
  match_level = TRUE,
  B = 500,
  top = 10,
  min_ratio = 3
)
```

Arguments

`data`	A data frame with one row per respondent.
`items`	Item columns. By default, the items in `scales` or every numeric column except `group`.
`group`	Optional batch of each respondent (a column name of `data` or a vector). Without it, the whole data set is compared with its shuffled version only.
`focal`	Group (or groups) under examination. By default, the smallest group.

`reference` Group (or groups) used as reference. By default, every other group.

`drop_flat` Logical; remove respondents with the same answer in every item before comparing.

`match_level` Logical; add the comparison with reference samples paired by level with the focal batch.

`B` Number of shuffles and of control samples.

`top` Number of closest pairs of the focal batch returned.

`min_ratio` Minimum ratio between the reference and the focal group for the whole batch to be compared in every sample. With a smaller reference (even one smaller than the batch) the batch is subsampled to `floor(n_reference / min_ratio)` respondents. Larger values make the control vary more but leave smaller focal samples, with less power. Use `1` for the behaviour of version 0.4.0 when the reference is at least as large as the batch.

Value

An object of class `neighbor_test`, a list with:

table Respondents, median nearest-neighbour distance (real and shuffled), their gap and the p value against the shuffles; with `group`, also the median gap of the control samples and the p value against them (share of control gaps as large or larger). When the batch was subsampled, `focal_gap_sample` is the median gap of the focal subsamples, the value compared with the control.

distances One row per examined respondent: row, distance to the nearest neighbour and the row of that neighbour.

pairs The `top` closest pairs of rows and the items in which they differ.

null, settings Values of the shuffles and control samples and the arguments used, kept for the plots.

See Also

`order_test()` and `compare_batches()`.

Examples

```
nt <- neighbor_test(batch_sim, items = paste0("I", 1:12), group = "phase", B = 100)
nt
plot(nt)
plot(nt, type = "distance")
```

`neighbor_test-methods` *Print and plot a near-duplicate test*

Description

`print()` shows the medians, the gap and the p values, and the closest pairs. `plot()` draws one of two views, with **ggplot2**:

`"null"` With `group`: histogram of the gap (shuffled minus real) in the control samples, with the focal gap as a vertical line. Without `group`: histogram of the shuffled medians with the real median.

`"distance"` Distribution of the nearest-neighbour distance in the examined group, in its shuffled version and in the control. Edited copies pile up at the left (one to three items of difference).

Usage

```
## S3 method for class 'neighbor_test'
print(x, ...)

## S3 method for class 'neighbor_test'
plot(
  x,
  type = c("null", "distance"),
  lang = c("en", "es"),
  base_size = 11,
  ...
)
```

Arguments

`x`	An object returned by `neighbor_test()`.
`...`	Not used.
`type`	`"null"` or `"distance"`.
`lang`	Language of the labels: `"en"` or `"es"`.
`base_size`	Base font size of the theme.

Value

`print()` returns `x` invisibly. `plot()` returns a `ggplot` object, printed on the current device and returned invisibly.

Examples

```
nt <- neighbor_test(batch_sim, items = paste0("I", 1:12), group = "phase", B = 100)
print(nt)
plot(nt, type = "distance", lang = "es")
```

`order_test` *Test whether consecutive rows are more alike than chance*

Description

When a block of answers is filled by hand, each row tends to be a copy of the previous one with a few changes. That leaves a trace that does not need the submission time: consecutive rows resemble each other more than rows taken in a random order. In genuine data the order of arrival is unrelated to the answers, so the observed order should not be special.

Usage

```
order_test(
  data,
  items = NULL,
  profile = NULL,
  group = NULL,
  time = NULL,
  B = 5000
)
```

Arguments

`data`	A data frame with one row per respondent, in the order of submission (or give `time`).
`items`	Item columns. By default, every numeric column except those named in `profile`, `group` and `time`.
`profile`	Optional character vector with categorical columns (for example demographic data) whose repetition in consecutive rows is tested as well.
`group`	Optional batch of each respondent (a column name or a vector). The test is run within each batch, in its own order.
`time`	Optional submission time (column name or vector) used to sort the rows; by default the rows are taken in the order given.
`B`	Number of random orders.

Details

For the items, the statistic is the mean number of items in which each row differs from the previous one (items answered in both). Its null distribution comes from `B` random orders of the same rows, and the p value is the share of orders with a mean as low or lower. For the `profile` columns (for example sex, age group and cycle), the statistic is the share of consecutive rows with exactly the same profile, and the p value is the share of random orders with a share as high or higher.

Respondents who answer together (the same classroom, the same session) can produce a mild order effect in genuine data. Use a group of known genuine data as a control: a batch is suspicious when its order effect is much stronger than the control's.

Value

An object of class `order_test`, a list with:

table One row per batch and statistic: respondents, observed value, mean of the random orders, p value.

consecutive One row per respondent after the first of its batch: original row, position, items that differ from the previous row and whether the profile is the same.

null, settings Values of the random orders and arguments used, kept for the plots.

See Also

`neighbor_test()` for near-duplicates in any order and `compare_batches()` for the correlation structure.

Examples

```
ot <- order_test(batch_sim, items = paste0("I", 1:12),
                 profile = c("sex", "age"), group = "phase", B = 500)
ot
plot(ot)
plot(ot, type = "sequence")
```

`order_test-methods` *Print and plot an order test*

Description

`print()` shows, for each batch, the observed and the expected values and the p value. `plot()` draws one of two views, with **ggplot2**:

`"null"` Histograms of the statistic in the random orders, with the observed order as a vertical line. Copying shows as a line far to the left (items) or to the right (profile) of the histogram.

`"sequence"` Items in which each row differs from the previous one, in order, with the value expected at random as a dashed line. A run of short bars marks a stretch of rows that resemble each other, the place to look first with the submission times.

Usage

```
## S3 method for class 'order_test'
print(x, ...)

## S3 method for class 'order_test'
plot(
  x,
  type = c("null", "sequence"),
  lang = c("en", "es"),
  base_size = 11,
  ...
)
```

Arguments

x	An object returned by order_test().
...	Not used.
type	"null" or "sequence".
lang	Language of the labels: "en" or "es".
base_size	Base font size of the theme.

Value

print() returns x invisibly. plot() returns a ggplot object, printed on the current device and returned invisibly.

Examples

```
ot <- order_test(batch_sim, items = paste0("I", 1:12), group = "phase", B = 500)
print(ot)
plot(ot, type = "sequence", lang = "es")
```

---

plot.response_scan *Plot a response screening*

Description

Ten views of the same screening, drawn with **ggplot2**:

"overview" Respondents flagged by each criterion. Bars do not add up: one person can cross several criteria.

"indices" Distribution of each index with its cut-off (dashed line). Look for tails or isolated bars far from the bulk.

"patterns" The most frequent response patterns; the dark part of each bar is the share that arrived in series. A very frequent pattern is suspicious only if it is implausible or arrives in series.

"heatmap" Every answer of every respondent (light = lowest option, dark = highest), with identical patterns grouped together and flagged respondents marked in red on the left. Blocks of identical rows reveal copied or repeated patterns.

"time" Minutes since the previous submission (log scale) along the collection period. Series appear as vertical piles of filled points below the dashed line.

"days" Submissions per day, split into series, other flagged respondents and the rest. Points to the session where something happened.

"overlap" How many respondents share each pair of flags (the diagonal is the total of each flag). Criteria that do not overlap detect different problems.

"profiles" Mean answer to each item (rescaled to 0-1, as answered) for series, other flagged respondents and the rest. A step between direct and reverse-worded items is the signature of a fixed pattern.

"styles" Proportion at the midpoint against proportion at the extremes, a map of response styles.

"scatter" Intra-individual variability against the Mahalanobis distance, with both cut-offs. Points in the left zone answer almost the same everywhere; points above the line answer unusually.

Usage

```
## S3 method for class 'response_scan'
plot(
  x,
  type = c("overview", "indices", "patterns", "heatmap", "time", "days", "overlap",
    "profiles", "styles", "scatter"),
  top = 10,
  lang = c("en", "es"),
  base_size = 11,
  ...
)
```

Arguments

x	An object returned by scan_responses().
type	One of "overview", "indices", "patterns", "heatmap", "time", "days", "overlap", "profiles", "styles" or "scatter".
top	Number of patterns shown by type = "patterns".
lang	Language of the labels: "en" (English) or "es" (Spanish).
base_size	Base font size of the theme.
...	Not used.

Value

A ggplot object, printed on the current device and returned invisibly, so it can be modified with further layers or saved with ggplot2::ggsave().

Examples

```
res <- scan_responses(survey_sim,
  scales = list(A = paste0("A", 1:8), B = paste0("B", 1:10)),
  reverse = paste0("A", 1:4), time = "submitted",
  pattern_items = paste0("A", 1:8))
plot(res)
plot(res, type = "indices")
plot(res, type = "patterns", top = 8, lang = "es")
plot(res, type = "heatmap")
plot(res, type = "time")
plot(res, type = "days")
plot(res, type = "overlap")
plot(res, type = "profiles")
plot(res, type = "styles")
g <- plot(res, type = "scatter")
g + ggplot2::labs(title = "IRV and Mahalanobis")
```

`response_scan-methods` *Print, summarise and convert a response screening*

Description

`print()` shows how many respondents each criterion flags, the most frequent pattern and the longest series. `summary()` returns the main tables. `as.data.frame()` returns one row per respondent with every index and flag, ready to be merged with the original data by position.

Usage

```
## S3 method for class 'response_scan'
print(x, top = 3, ...)

## S3 method for class 'response_scan'
summary(object, top = 10, ...)

## S3 method for class 'summary.response_scan'
print(x, ...)

## S3 method for class 'response_scan'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`, `object`	An object returned by `scan_responses()`.
`top`	Number of patterns or series to show.
`...`	Not used.
`row.names`, `optional`	Ignored; present for compatibility with the generic.

Value

`print()` returns `x` invisibly. `summary()` returns an object of class `summary.response_scan`, a list with `criteria` (one row per criterion with its rule, number and percentage of flagged respondents), `any` (respondents with at least one flag), `patterns` (the `top` most frequent patterns) and `series` (the `top` longest series). `as.data.frame()` returns a data frame with the columns of `x$indices` followed by the logical flags, prefixed with `flag_`, and `n_flags`.

Examples

```
res <- scan_responses(survey_sim, items = paste0("A", 1:8),
                      reverse = paste0("A", 1:4), time = "submitted")
summary(res)
out <- cbind(survey_sim, as.data.frame(res))
table(out$group, out$n_flags > 0)
```

scan_responses *Screen item responses for aberrant patterns*

Description

One call computes, for every respondent, the usual indices of careless or aberrant responding and flags the cases that cross each criterion. It also tabulates how often each exact response pattern occurs and, when the submission time is available, detects identical patterns that arrive in series of consecutive submissions (forms filled one after another, answers dictated or copied).

Usage

```
scan_responses(
  data,
  items = NULL,
  scales = NULL,
  reverse = NULL,
  range = NULL,
  time = NULL,
  pattern_items = NULL,
  min_run = 3,
  max_gap = 10,
  person_fit = FALSE,
  irv_quantile = 0.05,
  mahal_p = 0.001,
  syn_r = 0.6,
  syn_min_pairs = 5,
  dup_min_items = 15,
  flat_min_items = 6,
  eo_min_scales = 6,
  reference = NULL,
  ref_quantile = 0.05
)
```

Arguments

data	A data frame with one row per respondent.
items	Character vector with the item columns. By default, every numeric column of data (except time, when it is a column name).
scales	Optional named list of character vectors, one per scale or subscale. Flat responding is evaluated within each scale, and response ranges are inferred per scale. By default all items form one scale.
reverse	Optional character vector with the reverse-worded items. They are reverse-scored before the indices that assume a common direction (intra-individual variability, Mahalanobis distance, psychometric synonyms, person fit). Patterns, long strings and the heat map always use the responses as given.

`range` Optional numeric vector `c(min, max)` with the response options, applied to every item. By default it is taken from the minimum and maximum observed in each scale.

`time` Optional submission time: a column name of `data` or a vector of length `nrow(data)` (`POSIXct`, `Date`, character coercible to `POSIXct`, or numeric minutes).

`pattern_items` Items used to define a response pattern for the frequency table and the serial detection. By default, `items`. Use a single scale when the suspicion concerns that scale only.

`min_run` Minimum number of consecutive submissions with the same pattern to call them a series.

`max_gap` Maximum minutes between two consecutive submissions of a series.

`person_fit` Logical. If `TRUE`, fits a graded response model per scale with the **mirt** package and returns the standardized log-likelihood person-fit statistic Zh (Drasgow et al., 1985).

`irv_quantile` Respondents with some variability (IRV > 0) at or below this quantile of the non-zero values are flagged. Respondents with IRV = 0 answered the same option everywhere and are already covered by `flat`; leaving them out keeps the cut from collapsing to 0 when many people respond flat.

`mahal_p` Significance level for the Mahalanobis distance.

`syn_r` Minimum inter-item correlation for two items to be treated as psychometric synonyms.

`syn_min_pairs` Minimum number of disjoint synonym pairs (each item in one pair only, strongest pairs first) needed to evaluate `psychsyn`. With fewer pairs the within-person correlation rests on too few points and the criterion is reported as not evaluable. Short one-factor scales (fewer than `2 * syn_min_pairs` items) never reach it.

`dup_min_items` Minimum number of items for `duplicate` to be evaluated. With few items, two genuine respondents often give the same answers by chance (the probable patterns are few and the sample is large), so a shared vector is not a sign of copying. Vectors that are flat in every scale are not counted as duplicates: they are already flagged by `flat`.

`flat_min_items` Minimum number of items for a scale to be checked for flat responding. In a short scale (three or four items) many genuine respondents choose the same option everywhere. Shorter scales are skipped; when no scale is long enough, flat responding is judged on all items together.

`eo_min_scales` Minimum number of scales with two or more items needed to compute the even-odd consistency. Each scale is split into its odd- and even-positioned items, and the person's odd and even scale means are correlated across scales (Spearman-Brown corrected). With fewer scales the index rests on too few points and is not evaluated.

`reference` Optional rows that form a reference group of ordinary respondents: a logical vector of length `nrow(data)` or row positions. When given, the cut-offs of `low_irv`, `mahal`, `psychsyn` and `evenodd` are taken from the distribution of that group instead of fixed rules, and a respondent is flagged when more extreme than `ref_quantile` of the reference. Flat responders are left out of the reference, because they distort its distribution. Useful to judge a new batch of data against an earlier collection.

`ref_quantile` Tail probability used with `reference` (0.05: below the 5th percentile, or above the 95th for the Mahalanobis distance).

Value

An object of class `response_scan`, a list with:

indices Data frame, one row per respondent: `row` (position in `data`), `longstring` (longest run of identical consecutive answers), `irv` (intra-individual standard deviation after rescaling every item to 0-1), `mahal_d2` and `mahal_p`, `psychsyn` (within-person correlation across disjoint synonym pairs), `evenodd` (even-odd consistency, Spearman-Brown corrected), `midpoint` and `extreme` (proportion of answers at the midpoint and at either end), `pattern` and `pattern_n` (the response pattern and how many respondents share it), `series` (series identifier or `NA`) and, if requested, `zh`.

flags Data frame of logical flags: `flat`, `low_irv`, `mahal`, `psychsyn`, `evenodd`, `duplicate`, `serial` and, if requested, `zh`; plus `n_flags`.

patterns Data frame with every distinct pattern, its frequency, percentage and how many of its cases belong to a series, sorted by frequency.

series Data frame with one row per detected series (start, end, submissions, minutes, median gap and pattern); empty without `time`.

criteria Data frame describing each criterion, its rule and the number and percentage of flagged respondents.

responses, time, settings The item responses as given, the submission time and the arguments used, kept for the plots.

References

Curran, P. G. (2016). Methods for the detection of carelessly invalid responses in survey data. *Journal of Experimental Social Psychology, 66*, 4-19. doi:10.1016/j.jesp.2015.07.006

Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology, 38*(1), 67-86. doi:10.1111/j.20448317.1985.tb00817.x

Meade, A. W., & Craig, S. B. (2012). Identifying careless responses in survey data. *Psychological Methods, 17*(3), 437-455. doi:10.1037/a0028085

See Also

`plot.response_scan()` for the plots, `as.data.frame.response_scan()` to merge the indices back into the data, and `compare_batches()`, `order_test()` and `neighbor_test()` to judge a whole batch of data rather than single respondents.

Examples

```
res <- scan_responses(survey_sim,
  scales = list(A = paste0("A", 1:8), B = paste0("B", 1:10)),
  reverse = paste0("A", 1:4), time = "submitted",
  pattern_items = paste0("A", 1:8))
res
```

```
plot(res, type = "patterns")
plot(res, type = "heatmap")
head(as.data.frame(res))

# Cut-offs taken from an earlier collection (phase 1) to judge phase 2
res2 <- scan_responses(batch_sim, items = paste0("I", 1:12),
  reference = batch_sim$phase == 1)
res2$criteria
# One by one, the hand-filled rows look ordinary: they are found by
# judging the batch as a whole, with compare_batches(), order_test()
# and neighbor_test()
table(batch_sim$source, res2$flags$n_flags > 0)
```

`survey_sim` *Simulated survey with known aberrant respondents*

Description

Simulated answers of 300 respondents to two scales, with groups of aberrant respondents planted on purpose so that every criterion of `scan_responses()` has something to find. No real person is included.

Usage

```
survey_sim
```

Format

A data frame with 300 rows and 21 columns:

id Respondent number, in order of submission.

submitted Submission time (`POSIXct`, UTC).

A1 Scale A, item 1; three options (1 = No, 2 = More or less, 3 = Yes), reverse-worded.

A2 Scale A, item 2; three options (1 = No, 2 = More or less, 3 = Yes), reverse-worded.

A3 Scale A, item 3; three options (1 = No, 2 = More or less, 3 = Yes), reverse-worded.

A4 Scale A, item 4; three options (1 = No, 2 = More or less, 3 = Yes), reverse-worded.

A5 Scale A, item 5; three options (1 = No, 2 = More or less, 3 = Yes).

A6 Scale A, item 6; three options (1 = No, 2 = More or less, 3 = Yes).

A7 Scale A, item 7; three options (1 = No, 2 = More or less, 3 = Yes).

A8 Scale A, item 8; three options (1 = No, 2 = More or less, 3 = Yes).

B1 Scale B, item 1; five options (1 to 5).

B2 Scale B, item 2; five options (1 to 5).

B3 Scale B, item 3; five options (1 to 5).

B4 Scale B, item 4; five options (1 to 5).

B5 Scale B, item 5; five options (1 to 5).

B6 Scale B, item 6; five options (1 to 5).

B7 Scale B, item 7; five options (1 to 5).

B8 Scale B, item 8; five options (1 to 5).

B9 Scale B, item 9; five options (1 to 5).

B10 Scale B, item 10; five options (1 to 5).

group Planted group: `"typical"`, `"flat"` (same option everywhere), `"random"`, `"duplicate"` (copy of another respondent) or `"serial"` (the same extreme pattern in scale A, submitted one to two minutes apart).

Source

Simulated with the script `data-raw/make_survey_sim.R` of the package source.

Index

∗ **datasets**
batch_sim, 3
survey_sim, 22

as.data.frame.response_scan *(*response_scan-methods*)*, 18
as.data.frame.response_scan(), *21*

batch_comparison-methods, 2
batch_sim, 3

compare_batches, 4
compare_batches(), *3*, *11*, *12*, *15*, *21*
compare_fit, 7
compare_fit(), *10*

exclude_responses, 8
exclude_responses(), *7*

fit_comparison-methods, 10

ggplot2::ggsave(), *17*

neighbor_test, 11
neighbor_test(), *3*, *7*, *13*, *15*, *21*
neighbor_test-methods, 13

order_test, 14
order_test(), *3*, *7*, *12*, *16*, *21*
order_test-methods, 15

plot.batch_comparison *(*batch_comparison-methods*)*, 2
plot.fit_comparison *(*fit_comparison-methods*)*, 10
plot.neighbor_test *(*neighbor_test-methods*)*, 13
plot.order_test *(*order_test-methods*)*, 15
plot.response_scan, 16
plot.response_scan(), *21*
print.batch_comparison *(*batch_comparison-methods*)*, 2
print.fit_comparison *(*fit_comparison-methods*)*, 10
print.neighbor_test *(*neighbor_test-methods*)*, 13
print.order_test *(*order_test-methods*)*, 15
print.response_scan *(*response_scan-methods*)*, 18
print.summary.response_scan *(*response_scan-methods*)*, 18

response_scan-methods, 18

scan_responses, 19
scan_responses(), *7*, *9*, *17*, *18*, *22*
summary.response_scan *(*response_scan-methods*)*, 18
survey_sim, 22