Package 'PTSDdiag'

August 4, 2026

Title Optimize PTSD Diagnostic Criteria

Version 0.5.0

Description Provides tools for analyzing and optimizing PTSD (Post-Traumatic Stress Disorder) diagnostic criteria using PCL-5 (PTSD Checklist for DSM-5) and CAPS-5 (Clinician-Administered PTSD Scale for DSM-5) data. Functions identify optimal subsets of PCL-5 items that maintain diagnostic accuracy while reducing assessment burden. Includes tools for both hierarchical (cluster-based) and non-hierarchical symptom combinations, calculation of diagnostic metrics, and comparison with standard DSM-5 criteria. Redundancy analysis quantifies how many item subsets perform equivalently, through plateau sizes and bootstrap stability, and reports item selection against the chance baseline of the searched candidate space. A transport layer allows a site to evaluate subsets derived elsewhere and return only aggregate summaries, supporting multi-site validation without sharing individual-level data. Model validation is conducted using holdout and cross-validation methods to assess robustness and generalizability of the results. For more details see Weidmann et al. (2025) <doi:10.31219/osf.io/6rk72_v1>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports cli, dplyr, jsonlite, lifecycle, magrittr, rlang, rsample, stats, utils

Depends R (>= 3.5.0)

Suggests DT, future.apply, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

URL https://tobiasrspiller.github.io/PTSDdiag/, https://github.com/TobiasRSpiller/PTSDdiag

BugReports https://github.com/TobiasRSpiller/PTSDdiag/issues

NeedsCompilation no

Author Laura Weidmann [aut] (ORCID: <https://orcid.org/0009-0006-5509-6961>),
Tobias R. Spiller [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0107-0743>),
Flavio A. Schüepp [aut] (ORCID:
<https://orcid.org/0000-0002-0750-3888>)

Maintainer Tobias R. Spiller <tobias.spiller@access.uzh.ch>

Repository CRAN

Date/Publication 2026-08-04 16:30:28 UTC

Contents

`analyze_best_six_symptoms_four_required`

Find optimal non-hierarchical six-symptom combinations for PTSD diagnosis

---

Description

‘r lifecycle::badge("deprecated")‘

Convenience wrapper around `optimize_combinations` with the original PCL-5 defaults: 6 symptoms, 4 required, top 3 returned.

Identifies the three best six-symptom combinations for PTSD diagnosis where any four symptoms must be present, regardless of their cluster membership.

Usage

```
analyze_best_six_symptoms_four_required(
  data,
  score_by = "balanced_accuracy",
  DT = FALSE
)
```

Arguments

`data`	A dataframe containing exactly 20 columns with PCL-5 item scores (output of rename_ptsd_columns). Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely
`score_by`	Character string specifying optimization criterion: • "balanced_accuracy": Maximise balanced accuracy, the mean of sensitivity and specificity. Robust when one diagnostic class is much more common than the other. Default. • "accuracy": Minimize total misclassifications (FP + FN, i.e. maximise overall accuracy). • "sensitivity": Minimize false negatives only (i.e. maximise sensitivity relative to the full DSM-5-TR diagnosis).
`DT`	Logical. If `TRUE`, return the summary as an interactive `datatable` widget. If `FALSE` (default), return a plain data.frame.

Details

The function:

1. Tests all possible combinations of 6 symptoms from the 20 PCL-5 items
2. Requires 4 symptoms to be present (>=2 on original 0-4 scale) for diagnosis
3. Identifies the three combinations that best match the original DSM-5 diagnosis

Optimization can be based on:

- Maximizing balanced accuracy, the mean of sensitivity and specificity (the default)
- Minimizing false cases (both false positives and false negatives)
- Minimizing only false negatives (newly non-diagnosed cases)

The symptom clusters in PCL-5 are:

- Items 1-5: Intrusion symptoms (Criterion B)
- Items 6-7: Avoidance symptoms (Criterion C)
- Items 8-14: Negative alterations in cognitions and mood (Criterion D)
- Items 15-20: Alterations in arousal and reactivity (Criterion E)

Value

A list containing:

- best_symptoms: List of three vectors, each containing six symptom numbers representing the best combinations found
- diagnosis_comparison: Dataframe comparing original DSM-5 diagnosis with diagnoses based on the three best combinations
- summary: Diagnostic accuracy metrics for each combination. A data.frame by default, or an interactive `datatable` if `DT = TRUE`.

See Also

`optimize_combinations` for the generalized version with configurable parameters.

Examples

```
# Use a 250-row subset of the bundled data to keep the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))


# Find best combinations with the default balanced-accuracy criterion
results <- analyze_best_six_symptoms_four_required(ptsd_data)

# Get symptom numbers
results$best_symptoms

# View raw comparison data
results$diagnosis_comparison

# View summary statistics
results$summary
```

`analyze_best_six_symptoms_four_required_clusters` *Find optimal hierarchical six-symptom combinations for PTSD diagnosis*

Description

‘r lifecycle::badge("deprecated")‘

Convenience wrapper around `optimize_combinations_clusters` with the original PCL-5 defaults: 6 symptoms, 4 required, top 3 returned, and standard DSM-5 cluster structure.

Identifies the three best six-symptom combinations for PTSD diagnosis where four symptoms must be present and must include at least one symptom from each DSM-5 criterion cluster.

Usage

```
analyze_best_six_symptoms_four_required_clusters(
  data,
  score_by = "balanced_accuracy",
  DT = FALSE
)
```

Arguments

`data`	A dataframe containing exactly 20 columns with PCL-5 item scores (output of rename_ptsd_columns). Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely
`score_by`	Character string specifying optimization criterion: • "balanced_accuracy": Maximise balanced accuracy, the mean of sensitivity and specificity. Robust when one diagnostic class is much more common than the other. Default. • "accuracy": Minimize total misclassifications (FP + FN, i.e. maximise overall accuracy). • "sensitivity": Minimize false negatives only (i.e. maximise sensitivity relative to the full DSM-5-TR diagnosis).
`DT`	Logical. If `TRUE`, return the summary as an interactive `datatable` widget. If `FALSE` (default), return a plain data.frame.

Details

The function:

1. Generates valid combinations ensuring representation from all clusters
2. Requires 4 symptoms to be present (>=2 on original 0-4 scale) for diagnosis
3. Validates that present symptoms include at least one from each cluster
4. Identifies the three combinations that best match the original DSM-5 diagnosis

DSM-5 PTSD symptom clusters:

- Cluster 1 (B) - Intrusion: Items 1-5
- Cluster 2 (C) - Avoidance: Items 6-7
- Cluster 3 (D) - Negative alterations in cognitions and mood: Items 8-14
- Cluster 4 (E) - Alterations in arousal and reactivity: Items 15-20

Optimization can be based on:

- Maximizing balanced accuracy, the mean of sensitivity and specificity (the default)
- Minimizing false cases (both false positives and false negatives)
- Minimizing only false negatives (newly non-diagnosed cases)

Value

A list containing:

- best_symptoms: List of three vectors, each containing six symptom numbers representing the best combinations found
- diagnosis_comparison: Dataframe comparing original DSM-5 diagnosis with diagnoses based on the three best combinations
- summary: Diagnostic accuracy metrics for each combination. A data.frame by default, or an interactive `datatable` if `DT = TRUE`.

See Also

`optimize_combinations_clusters` for the generalized version with configurable parameters and custom cluster definitions.

Examples

```
# This deprecated wrapper always runs the full 6-symptom hierarchical
# search, so its example uses a 50-row subset to stay fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:50, ],
                                 id_col = c("patient_id", "age", "sex"))


# Find best hierarchical combinations with the default balanced-accuracy
# criterion
results <- analyze_best_six_symptoms_four_required_clusters(ptsd_data)

# Get symptom numbers
results$best_symptoms

# View raw comparison data
results$diagnosis_comparison

# View summary statistics
results$summary
```

---

apply_symptom_combinations

Apply pre-specified symptom combinations to new data

Description

Applies one or more pre-specified symptom combinations to a new dataset and returns a comparison dataframe with the baseline DSM-5 diagnosis and the simplified diagnosis for each combination. This function enables external validation by allowing combinations derived from one dataset to be tested on another.

Usage

```
apply_symptom_combinations(data, combinations, n_required = 4, clusters = NULL)
```

Arguments

`data`	A dataframe containing exactly 20 columns with PCL-5 item scores (output of `rename_ptsd_columns`). Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely
`combinations`	A list of integer vectors. Each vector contains symptom indices (e.g., `c(4, 6, 7, 17, 19, 20)`). All vectors must have the same length. Typically obtained from `results$best_symptoms` of an optimization run.
`n_required`	Integer specifying how many symptoms must be present (scored >= 2 on original scale) for a positive diagnosis (default: 4). Must be between 1 and the length of each combination.
`clusters`	`NULL` (default) for non-hierarchical checking, or a named list of integer vectors defining the cluster structure for hierarchical checking. When provided, a positive diagnosis additionally requires that the present symptoms span all defined clusters. For PCL-5: `list(B = 1:5, C = 6:7, D = 8:14, E = 15:20)`

Details

The function:

1. Computes the baseline DSM-5 diagnosis (`PTSD_orig`) on the provided data using binarized scores
2. Binarizes the symptom data (scores >= 2 become 1, others become 0)
3. For each combination, determines diagnosis based on whether at least `n_required` symptoms are present
4. If `clusters` is provided, additionally checks that present symptoms span all defined clusters (hierarchical checking)
5. Returns a comparison dataframe suitable for `summarize_ptsd_changes`

Typical workflow for external validation:

1. Derive optimal combinations on Dataset A using `optimize_combinations` or `optimize_combinations_clusters`
2. Apply those combinations to Dataset B using this function
3. Compute diagnostic metrics using `summarize_ptsd_changes` and `create_readable_summary`

Value

A dataframe with columns:

- Any carry-through columns from `data` (e.g. an ID column added via `rename_ptsd_columns`), prepended in original order.
- `PTSD_orig`: Logical. Full DSM-5 diagnosis computed on this data.
- One column per combination (logical): Simplified diagnosis for each combination. Column names follow the `symptom_X_Y_Z` pattern.

This dataframe can be passed directly to `summarize_ptsd_changes` to compute diagnostic accuracy metrics.

See Also

`optimize_combinations` and `optimize_combinations_clusters` for deriving optimal combinations.

`summarize_ptsd_changes` and `create_readable_summary` for computing and formatting diagnostic metrics from the returned comparison dataframe.

Examples

```
# Create example data
set.seed(42)
ptsd_data <- data.frame(matrix(sample(0:4, 400, replace=TRUE), ncol=20))
names(ptsd_data) <- paste0("symptom_", 1:20)

# Define some combinations to test
my_combinations <- list(
  c(1, 6, 8, 10, 15, 19),
  c(2, 7, 9, 11, 16, 20)
)

# Apply combinations (non-hierarchical)
comparison <- apply_symptom_combinations(ptsd_data, my_combinations,
                n_required = 4)

# Compute metrics
metrics <- summarize_ptsd_changes(comparison)
create_readable_summary(metrics)

# Apply with hierarchical checking
pcl5_clusters <- list(B = 1:5, C = 6:7, D = 8:14, E = 15:20)
hier_comparison <- apply_symptom_combinations(ptsd_data, my_combinations,
                     n_required = 4, clusters = pcl5_clusters)
summarize_ptsd_changes(hier_comparison)
```

`as_definitions` *Convert imported combination specifications into definitions*

Description

Converts one or more combination specifications – as returned by `read_combinations` – into the definitions list that `evaluate_definitions` expects. This is the validation-site counterpart of `extract_definitions`: the derivation site exports each rule as a JSON file, and the validation site turns the imported files into evaluable definitions in one call.

Usage

```
as_definitions(specs, n_top = NULL)
```

Arguments

`specs`	A single combination specification (the list returned by `read_combinations`, or any list with `combinations` and `n_required` elements), or a list of such specifications – e.g. `lapply(files, read_combinations)`. List names, when supplied, become the definition labels.
`n_top`	Integer or `NULL` (default). When supplied, only the first `n_top` combinations of each specification (their stored rank order) are kept; capped at the number available.

Details

Each definition is labelled by, in order of precedence: the name given to the list element in `specs`; the `label` stored in the file by `write_combinations`; or an automatic label of the form `"4/6 Hierarchical"` derived from the rule (`n_required` / number of symptoms, hierarchical when the specification carries a cluster structure). Duplicate labels are an error – name the list elements to disambiguate.

A cluster structure stored in the specification is preserved in the definition, so hierarchical rules exported with non-default clusters are evaluated with exactly those clusters.

Value

A named list in the shape returned by `extract_definitions`: one element per specification, each a list with `symptoms` (list of integer vectors), `n_required`, `hierarchical`, and `clusters` (`NULL` unless the specification stored a cluster structure). Pass it to `evaluate_definitions`.

See Also

`read_combinations`, `evaluate_definitions`, `extract_definitions`.

Examples

```
# A derivation site exports a rule ...
tmp <- tempfile(fileext = ".json")
write_combinations(list(c(1, 6, 8, 10, 15, 19), c(2, 7, 9, 11, 16, 20)),
                   tmp, n_required = 4, label = "4/6 Non-hierarchical")

# ... and the validation site turns the imported file into definitions
spec <- read_combinations(tmp)
definitions <- as_definitions(spec, n_top = 2)
str(definitions)

unlink(tmp)
```

`as_set_matrix` *Coerce a combination specification to an integer matrix*

Description

Accepts the three shapes combinations arrive in: a character vector of combination ids, a list of integer vectors, or a `ptsdiag_spec` from `read_combinations`.

Usage

```
as_set_matrix(sets)
```

Arguments

`sets`	A combination specification.

Value

A list with `mat`, an integer matrix of one row per combination, and `ids`, the corresponding combination ids.

Examples

```
as_set_matrix(c("1_2_3_4_5_6", "2_3_6_7_17_18"))
```

`ba_ci` *Interval for balanced accuracy*

Description

Balanced accuracy is the mean of two proportions estimated on disjoint groups, so sensitivity and specificity are independent. Two ways of using that are offered, and the difference matters at the boundary.

Usage

```
ba_ci(tp, fn, fp, tn, conf_level = 0.95, method = c("wilson", "exact", "wald"))
```

Arguments

`tp`, `fn`, `fp`, `tn`	Counts of true positives, false negatives, false positives and true negatives. Vectorised and recycled together.
`conf_level`	Confidence level.
`method`	`"wilson"` (default) or `"exact"` to square-and-add the corresponding component intervals, or `"wald"` for the analytic normal-approximation variance.

Details

`method = "wilson"` (default) and `method = "exact"` combine the component intervals by Newcombe's square-and-add: with component intervals (l_1, u_1) and (l_2, u_2) around p_1 and p_2,

$$\text{lo} = \text{BA} - \tfrac{1}{2}\sqrt{(p_1 - l_1)^2 + (p_2 - l_2)^2}$$
$$\text{hi} = \text{BA} + \tfrac{1}{2}\sqrt{(u_1 - p_1)^2 + (u_2 - p_2)^2}$$

`method = "wald"` uses the analytic normal-approximation form $\text{Var}(\text{BA}) = \frac{1}{4}(\text{Var}(\text{sens}) + \text{Var}(\text{spec}))$.

The Wald form is degenerate at a boundary. $\text{Var}(p) = p(1-p)/n$ is zero when a proportion is 0 or 1, so a component at the boundary contributes no uncertainty at all. A rule with perfect sensitivity on six positive cases and perfect specificity on 177 negatives returns `1.00 (1.00, 1.00)` — an interval of zero width from 183 observations. The square-and-add methods do not have this failure, because Wilson and Clopper-Pearson intervals are not degenerate at the boundary: the same table gives `1.00 (0.77, 1.00)` under `"exact"`. `ba_ci()` warns if `"wald"` is used on a boundary proportion.

Whichever method is used, the interval is **marginal**. It is not a valid basis for comparing two symptom combinations, because such a comparison is paired on the same individuals and the two estimates are strongly positively correlated. Any table printing it alongside several combinations should say so.

Value

A three-column matrix with columns `est`, `lo` and `hi`, on the 0-1 scale.

References

Newcombe RG (1998). Interval estimation for the difference between independent proportions: comparison of eleven methods. *Statistics in Medicine*, 17(8), 873-890.

Examples

```
ba_ci(tp = 80, fn = 20, fp = 10, tn = 90)

# At the boundary the analytic form collapses and the others do not.
ba_ci(tp = 6, fn = 0, fp = 0, tn = 177, method = "exact")
suppressWarnings(ba_ci(tp = 6, fn = 0, fp = 0, tn = 177, method = "wald"))
```

---

binarize_data *Binarize PCL-5 symptom scores*

Description

Converts PCL-5 symptom scores from their original 0-4 scale to binary values (0/1) based on the clinical threshold for symptom presence (>=2).

Usage

```
binarize_data(data)
```

Arguments

data	A dataframe containing exactly 20 columns with PCL-5 item scores (output of rename_ptsd_columns). Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely Note: This function should only be used with raw symptom scores before calculating the total score, as it will convert all values in the dataframe to 0/1, which would invalidate any total score column if present.

Details

The function implements the standard clinical threshold for PTSD symptom presence where:

- Scores of 0-1 ("Not at all" and "A little bit") $\rightarrow$ 0 (symptom absent)
- Scores of 2-4 ("Moderately" to "Extremely") $\rightarrow$ 1 (symptom present)

Value

A dataframe with the same structure as input but with all symptom scores converted to binary values:

- 0 = Symptom absent (original scores 0-1)
- 1 = Symptom present (original scores 2-4)

Examples

```
# Create sample data
sample_data <- data.frame(
  matrix(sample(0:4, 20 * 10, replace = TRUE),
         nrow = 10,
         ncol = 20)
)
colnames(sample_data) <- paste0("symptom_", 1:20)

# Binarize scores
binary_data <- binarize_data(sample_data)
binary_data # Should only show 0s and 1s
```

---

bootstrap_stability *Bootstrap stability of near-optimal symptom combinations*

Description

Repeats the entire exhaustive search on bootstrap resamples of the respondents, and reports how often each symptom combination stays on the plateau and how often it is the outright winner. This answers the question a single search cannot: is the best-performing combination genuinely best, or did it win by a margin that would not survive a different sample of patients?

Usage

```
bootstrap_stability(
  fit,
  data,
  delta = 1,
  n_boot = 2000,
  seed = 123,
  chunk_size = 2000,
  show_progress = TRUE
)
```

Arguments

fit	A fitted exhaustive search from score_all_combinations.
data	The data frame that was passed to score_all_combinations, with the 20 PCL-5 item columns symptom_1 through symptom_20.
delta	Numeric. Plateau width in **percentage points** of balanced accuracy (default 1). A single value.
n_boot	Integer. Number of bootstrap replicates (default 2000).
seed	Integer. Random seed. The function restores the calling session's random number state when it exits.
chunk_size	Integer. Number of combinations processed per block (default 2000). Affects speed and memory only, never the result.
show_progress	Logical. If TRUE (default), report a runtime estimate and display a progress bar.

Details

Individuals are resampled with replacement, never combinations. In each replicate the balanced accuracy of every combination is recomputed and the reference point BA_best(b) is taken as the maximum *within that replicate*. This keeps every comparison on the scale of its own replicate: a resampled balanced accuracy is measured against a maximum drawn from the same resample, not against the original sample's maximum, which resamples would fall short of for reasons having nothing to do with the combination.

Read pi with the selection in mind. For a combination fixed in advance it estimates how often that combination is near-optimal in samples like this one. For the combination that happened to win in *this* sample it is optimistic, because bootstrap replicates share most of their observations with the sample that singled it out; treat the winner's pi as an upper bound rather than an out-of-sample rate. The plateau size, which does not depend on which combination was selected, is not affected by this.

The computation is arranged as matrix products rather than a loop over combinations. Because the reference diagnosis and every decision rule depend on the data only through the binarized responses (item score >= 2), respondents are first collapsed to unique response patterns. Resampling individuals is then equivalent to drawing pattern counts from a multinomial distribution, and the counts of true and false positives for every combination in every replicate follow from two matrix multiplications. A full 2000-replicate run over all 38,760 six-item combinations takes minutes rather than days.

Before resampling begins, the function recomputes balanced accuracy for a spread of combinations from data and checks the result against fit. This catches a mismatched data argument as well as any disagreement between the matrix path and the package's per-row rule.

Value

An object of class ptsdiag_stability: a list of three data frames.

- per_set: one row per combination, with combination_id, rank and balanced_accuracy carried over from fit, pi (proportion of replicates in which the combination lay within delta of that replicate's best), p_argmax (proportion of replicates in which it was the best, ties shared equally), n_argmax_outright and n_argmax_tied.

- `per_replicate`: one row per replicate, with `replicate`, `ba_best`, `plateau_size`, `plateau_proportion`, `n_tied_best`, `n_cases` and `n_noncases`.
- `summary`: one row, with `delta`, `n_boot`, `n_boot_used`, `n_candidates`, and the mean and 2.5th / 97.5th percentiles of the plateau size and proportion across replicates.

See Also

`compute_plateau`, `symptom_selection`, `compare_rule_forms`.

Examples

```
# A compact 4-symptom search and 50 replicates keep the example fast;
# a published analysis would use the full search and n_boot = 2000
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:120, ],
                                 id_col = c("patient_id", "age", "sex"))
fit <- score_all_combinations(ptsd_data, n_symptoms = 4, n_required = 3,
                              show_progress = FALSE)

stability <- bootstrap_stability(fit, ptsd_data, n_boot = 50,
                                 show_progress = FALSE)
stability$summary
head(stability$per_set)
```

`calculate_ptsd_total` *Calculate PTSD total score*

Description

Calculates the total PCL-5 (PTSD Checklist for DSM-5) score by summing all 20 symptom scores. The total score ranges from 0 to 80, with higher scores indicating greater symptom severity.

Usage

```
calculate_ptsd_total(data)
```

Arguments

`data`	A dataframe containing standardized PCL-5 item scores (output of rename_ptsd_columns). Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely

Details

Calculates the total score from PCL-5 items

Value

A dataframe with all original columns plus an additional column "total" containing the sum of all 20 symptom scores (range: 0-80)

Examples

```
# Create sample data
sample_data <- data.frame(
  matrix(sample(0:4, 20 * 10, replace = TRUE),
         nrow = 10,
         ncol = 20)
)
colnames(sample_data) <- paste0("symptom_", 1:20)

# Calculate total scores
scores_with_total <- calculate_ptsd_total(sample_data)
print(scores_with_total$total)
```

---

chance_baseline *Chance baseline for each item*

Description

The proportion of a rule form's candidate space that contains each item – how often an item would be selected if selection were arbitrary. Enumerated from the candidate space rather than assumed, which matters under a cluster constraint: a two-item cluster contributes far more of its combinations than a seven-item cluster, so the baseline is a step function of the cluster rather than a constant.

Usage

```
chance_baseline(n_symptoms = 6L, clusters = NULL, n_total = 20L)
```

Arguments

`n_symptoms`	Combination size.
`clusters`	`NULL`, or a named list of integer vectors.
`n_total`	Number of items to choose from.

Details

The baselines sum to `n_symptoms` over the items, by construction.

Value

A numeric vector of length `n_total`, on the 0-1 scale.

Examples

```
# A small space keeps the example instant; the pattern is the same as at six
# symptoms, with the two-item cluster far above the seven-item one. Four is
# the smallest size that can cover four clusters.
round(chance_baseline(3) * 100, 1)
clusters <- list(B = 1:5, C = 6:7, D = 8:14, E = 15:20)
round(chance_baseline(4, clusters) * 100, 1)
```

`check_pcl5_data` *Check PCL-5 item data before starting the workflow*

Description

Pre-flight check for a data frame of PCL-5 item scores. Unlike the fail-fast validation inside the workflow functions, this check runs **all** checks and reports every problem at once – column count, numeric type, the integer 0-4 scoring range, and missing values – so a data file can be fixed in one pass instead of one error at a time. Run it on your item columns before `rename_ptsd_columns`.

Usage

```
check_pcl5_data(data, id_col = NULL)
```

Arguments

`data`	A data frame containing the PCL-5 item columns (plus any columns named in `id_col`).
`id_col`	Optional character vector naming identifier column(s) to exclude from the check, mirroring `rename_ptsd_columns`.

Details

Two input shapes are supported. If `data` already contains the renamed columns `symptom_1` through `symptom_20`, those are checked by name (extra columns such as `total` are ignored). Otherwise all non-`id_col` columns are treated as the item columns in positional DSM-5 order, exactly as `rename_ptsd_columns` will interpret them – the check then also requires that there are exactly 20 of them.

Rows in which every item is 0 are reported as an informational note, not an error: whether symptom-free records are kept (e.g. to contribute true negatives in a validation sample) or excluded is an analytic choice.

Value

`invisible(TRUE)` when every check passes; otherwise the function aborts with a report listing all failed checks.

See Also

rename_ptsd_columns, which this check prepares for.

Examples

```
# Clean data passes and reports each check
ptsd_items <- simulated_ptsd[1:100, paste0("S", 1:20)]
check_pcl5_data(ptsd_items)

# A data frame with several problems reports them all in one error
bad <- ptsd_items
bad$extra_column <- 1          # 21 item columns
bad$S3[5] <- NA                # a missing value
bad$S7[2] <- 9                 # out of the 0-4 range
try(check_pcl5_data(bad))
```

---

compare_diagnostic_systems

Compare multiple diagnostic systems against a reference standard

---

Description

Produces a single unified summary table comparing the diagnostic performance of multiple criteria against a chosen reference standard. Suitable for use as a manuscript table comparing optimized symptom combinations, ICD-11, CAPS-5, and DSM-5-TR in one kable-ready output.

Usage

```
compare_diagnostic_systems(
  data,
  ...,
  icd11 = TRUE,
  caps5_data = NULL,
  reference = c("pcl5", "caps5"),
  labels = NULL
)
```

Arguments

data	A dataframe containing exactly 20 columns of PCL-5 item scores (output of rename_ptsd_columns). Always required. Used to compute the PCL-5 DSM-5-TR diagnosis and, when icd11 = TRUE, the ICD-11 diagnosis.
...	Zero or more comparison dataframes, each containing a PTSD_orig column and at least one additional logical column representing a diagnostic system (e.g. output of apply_symptom_combinations). When caps5_data is NULL, all PTSD_orig columns must be identical to the one computed from data. When caps5_data is provided, only row counts are validated.

icd11	Logical. If TRUE (default), compute the ICD-11 PTSD diagnosis from data and include it as a row in the output.
caps5_data	Optional dataframe containing exactly 20 columns of CAPS-5 item severity scores (output of rename_caps5_columns). Must have the same number of rows as data (paired participants). When provided, the CAPS-5 DSM-5-TR diagnosis is computed internally and included in the comparison.
reference	Character. Which DSM-5-TR diagnosis serves as the reference standard: "pcl5" (default) or "caps5". The reference row always has sensitivity = specificity = 1 and zero misclassifications. Setting reference = "caps5" requires caps5_data to be provided.
labels	Optional character vector of display names for the systems coming from ..., in the order the columns appear across all ... inputs (excluding PTSD_orig columns). Does not apply to built-in rows (DSM-5-TR, ICD-11, CAPS-5), which are always labelled automatically. If NULL (default), column names are used. A warning is issued if the length does not match.

Details

The function:

1. Computes the PCL-5 DSM-5-TR diagnosis from data
2. If caps5_data is provided, computes the CAPS-5 DSM-5-TR diagnosis
3. Sets the reference standard based on reference: either the PCL-5 or CAPS-5 DSM-5-TR diagnosis. The reference row always appears first with sensitivity = specificity = 1.
4. Optionally computes ICD-11 diagnosis from data when icd11 = TRUE
5. Collects all non-PTSD_orig columns from the ... comparison dataframes (e.g. output of apply_symptom_combinations)
6. Calls summarize_ptsd_changes internally and reshapes the result into a presentation-ready table

When caps5_data is NULL (default), labels follow the original convention: "DSM-5-TR" and "ICD-11". When caps5_data is provided, labels are disambiguated with the instrument name: "DSM-5-TR (PCL-5)", "DSM-5-TR (CAPS-5)", "ICD-11 (PCL-5)".

When caps5_data is provided, the strict PTSD_orig validation on ... inputs is relaxed to a row-count check only, because comparison dataframes may have been derived from either the PCL-5 or CAPS-5 data (which produce different PTSD_orig vectors).

Value

A data.frame with one row per diagnostic system and the following columns:

- system: Display name of the diagnostic criterion
- n_diagnosed: Number of cases meeting the criterion
- pct_diagnosed: Percentage of total sample diagnosed (2 dp)
- sensitivity: 4 dp
- specificity: 4 dp

- ppv: Positive predictive value, 4 dp
- npv: Negative predictive value, 4 dp
- n_false_negative: Cases missed vs. reference
- pct_false_negative: Percentage of total sample, 2 dp
- n_false_positive: Cases over-diagnosed vs. reference
- pct_false_positive: Percentage of total sample, 2 dp
- n_misclassified: Total misclassified cases
- accuracy: Proportion classified the same as the reference ((total - misclassified) / total), 4 dp
- balanced_accuracy: Mean of sensitivity and specificity ((sensitivity + specificity) / 2), 4 dp

See Also

create_icd11_diagnosis for the ICD-11 comparison dataframe.

create_caps5_diagnosis for standalone CAPS-5 diagnosis.

apply_symptom_combinations for generating comparison dataframes from optimized symptom combinations.

optimize_combinations and optimize_combinations_clusters for deriving optimal combinations.

Examples

```
ptsd_data <- rename_ptsd_columns(simulated_ptsd,
                                 id_col = c("patient_id", "age", "sex"))

# ICD-11 vs DSM-5-TR only (no optimized combinations)
tbl <- compare_diagnostic_systems(ptsd_data, icd11 = TRUE)
tbl


# Add two pre-specified combinations
combos <- apply_symptom_combinations(
  ptsd_data,
  combinations = list(c(1, 6, 8, 10, 15, 19), c(2, 7, 9, 11, 16, 20)),
  n_required = 4
)
tbl2 <- compare_diagnostic_systems(
  ptsd_data, combos,
  icd11  = TRUE,
  labels = c("Combo A", "Combo B")
)
knitr::kable(tbl2)

# With CAPS-5 as gold standard reference
caps5_raw <- data.frame(matrix(sample(0:4, 20 * nrow(simulated_ptsd),
                                      replace = TRUE), ncol = 20))
caps5_data <- rename_caps5_columns(caps5_raw)
tbl3 <- compare_diagnostic_systems(
  ptsd_data, combos,
```

```
  icd11      = TRUE,
  caps5_data = caps5_data,
  reference  = "caps5"
)
knitr::kable(tbl3)
```

`compare_optimizations` *Run multiple PTSD optimization scenarios in one call*

Description

Runs several optimization scenarios on the same dataset and bundles the results into a single object suitable for tabular and visual comparison. Reproduces the multi-scenario workflow used in the PTSDdiag preprint (4/6 hierarchical, 4/6 non-hierarchical, 3/6 non-hierarchical) in one call, and also supports adding fixed criteria such as ICD-11 to the comparison.

Usage

```
compare_optimizations(
  data,
  scenarios = NULL,
  include_icd11 = FALSE,
  n_top = 10,
  score_by = "balanced_accuracy",
  clusters = NULL,
  show_progress = TRUE
)
```

Arguments

`data`	A dataframe containing the 20 PCL-5 item columns `symptom_1` through `symptom_20` (output of `rename_ptsd_columns`). Non-symptom columns (e.g. a participant identifier) are carried through every scenario's per-row diagnosis output.
`scenarios`	Optional named list of scenario configurations. Each element is a list with: • `type`: `"optimize"` (default if omitted) or `"fixed"`. • For `type = "optimize"`: `n_symptoms` (integer 1-20), `n_required` (integer 1-`n_symptoms`), `hierarchical` (single logical), and optional `clusters` (named list of integer vectors; defaults to PCL-5 B/C/D/E when `hierarchical = TRUE`). • For `type = "fixed"`: `criterion` – either a known string (`"icd11"` or `"caps5"`) or a logical vector of length `nrow(data)` representing a pre-computed diagnosis. When supplying a logical vector you must also provide `symptoms`, the integer indices counted as "included" in the heatmap. When `NULL` (default), runs the three preprint scenarios: 4/6 hierarchical, 4/6 non-hierarchical, 3/6 non-hierarchical.

include_icd11	Logical. When TRUE, appends an "ICD-11" fixed-criterion scenario after any user-supplied entries (deduplicated by label). Default FALSE.
n_top	Integer. Number of top combinations to retain per optimize scenario (default 10). Fixed scenarios always contribute exactly one combination regardless of n_top.
score_by	Character. Optimization criterion: "balanced_accuracy" (maximise the mean of sensitivity and specificity), "accuracy" (minimise FP + FN), or "sensitivity" (minimise FN only). Applied to optimize scenarios that do not override it. Default "balanced_accuracy".
clusters	Optional named list of integer vectors defining the PCL-5 clusters used by hierarchical optimize scenarios that do not specify their own. Defaults to the DSM-5 B/C/D/E grouping when needed.
show_progress	Logical. Forwarded to each optimize scenario's progress bar. Default TRUE.

Details

Each scenario is either:

- **optimize** (default): runs optimize_combinations or optimize_combinations_clusters depending on hierarchical. Returns the top n_top combinations.
- **fixed**: applies a pre-defined diagnostic criterion (such as ICD-11 PTSD) and treats its fixed symptom set as a single "combination" for the purpose of the multi-scenario tables and heatmap.

Fixed scenarios let researchers benchmark optimized criteria against published systems in a uniform output.

Any non-symptom columns present in data (e.g. an ID column added via rename_ptsd_columns(..., id_col = "patient_id")) are carried through to each scenario's per-row diagnosis_comparison, so per-participant diagnoses can be joined back to demographics.

Value

An object of class ptsdiag_comparison, a list with:

- scenarios: named list of per-scenario results. Each element mirrors the shape returned by optimize_combinations (best_symptoms, diagnosis_comparison, summary, n_tied) and carries a type attribute.
- config: data.frame with one row per scenario summarising the configuration used.
- n_rows: number of input rows.
- call: the matched call.

Pass the result to summarize_top_combinations for a manuscript-ready performance table, to symptom_frequency for the long-format symptom inclusion counts, and to plot_symptom_frequency for the heatmap.

See Also

optimize_combinations, optimize_combinations_clusters, create_icd11_diagnosis, summarize_top_combinatio
symptom_frequency, plot_symptom_frequency.

Examples

```
# Use a 250-row subset of the bundled data to keep the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))

# A compact optimized rule plus ICD-11 (a small 4-symptom search keeps
# the example fast; omit `scenarios` to run the three default rules)
comp <- compare_optimizations(
  ptsd_data,
  scenarios = list(
    "3/4 Non-hierarchical" = list(n_symptoms = 4, n_required = 3,
                                  hierarchical = FALSE)
  ),
  include_icd11 = TRUE,
  n_top = 5,
  show_progress = FALSE
)
print(comp)

# Manuscript Table 2
summarize_top_combinations(comp, as_percent = TRUE)
```

`compare_rule_forms` *Compare plateaus and stability across rule forms*

Description

Runs the plateau analysis, and optionally the bootstrap stability analysis, across several exhaustive searches at once and stacks the results into long tables ready for reporting. Use it to put the hierarchical and non-hierarchical rules, or different endorsement thresholds, side by side.

Usage

```
compare_rule_forms(
  fits,
  data = NULL,
  delta = 1,
  n_boot = 0,
  seed = 123,
  show_progress = TRUE
)
```

Arguments

`fits`	A **named** list of `score_all_combinations` results. The names label the rule forms in the output, e.g. `list("4/6 Hierarchical" = fit_a, "3/6 Non-hierarchical" = fit_b)`.

data	The data frame the searches were run on. Required only when n_boot > 0.
delta	Numeric. Plateau width in **percentage points** of balanced accuracy (default 1). A vector is allowed, e.g. delta = c(1, 5); the bootstrap, which is far more expensive, uses the first value only.
n_boot	Integer. Bootstrap replicates per rule form. The default 0 skips the bootstrap entirely, so the plateau comparison stays cheap.
seed	Integer. Random seed shared by every rule form's bootstrap.
show_progress	Logical. If TRUE (default), report progress.

Details

Each rule form must be scored separately, because their candidate spaces differ: a six-item hierarchical search evaluates 13,685 combinations against the non-hierarchical search's 38,760. Every proportion in the output is divided by the denominator belonging to its own rule form, which is exactly the comparison a single pooled table would get wrong.

Value

An object of class ptsdiag_rule_forms: a list of long data frames, each carrying a rule_form column.

- plateau: the summary of compute_plateau for every rule form and delta.
- selection: symptom_selection for every rule form and delta, one row per item.
- stability, per_set: the summary and per_set tables of bootstrap_stability, present only when n_boot > 0.

The stability weighted columns of selection are filled only for the first delta, since that is the width the bootstrap ran at; rows for any further widths carry NA there rather than weights describing a different plateau.

See Also

compute_plateau, bootstrap_stability, symptom_selection.

Examples

```
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:120, ],
                                 id_col = c("patient_id", "age", "sex"))
fits <- list(
  "3/4 Non-hierarchical" = score_all_combinations(
    ptsd_data, n_symptoms = 4, n_required = 3, show_progress = FALSE
  ),
  "2/4 Non-hierarchical" = score_all_combinations(
    ptsd_data, n_symptoms = 4, n_required = 2, show_progress = FALSE
  )
)

comparison <- compare_rule_forms(fits, delta = c(1, 5))
comparison$plateau
```

compute_plateau *Find the plateau of near-optimal symptom combinations*

Description

Identifies every symptom combination whose balanced accuracy falls within `delta` percentage points of the best combination in the same search. The best-performing subset of PCL-5 items is rarely alone: typically a large set of alternatives performs indistinguishably well. Reporting the size of that plateau, rather than only the winner, is what separates "these six items are the right ones" from "any of these hundreds of six-item sets would do".

Usage

```
compute_plateau(fit, delta = 1)
```

Arguments

`fit`	A fitted exhaustive search from `score_all_combinations`.
`delta`	Numeric. Plateau width in **percentage points** of balanced accuracy: `delta = 1` (the default) keeps every combination within one percentage point of the best. A vector is allowed, e.g. `delta = c(1, 5)` to report a primary and a secondary plateau in one call.

Details

The plateau is always computed *within* one rule form, because the candidate spaces differ in size. A six-item hierarchical search evaluates 13,685 combinations while the non-hierarchical search evaluates 38,760, so the same plateau count means very different things; `plateau_proportion` divides by the right denominator for the search that produced `fit`.

Comparisons use exact integer arithmetic rather than the rounded balanced accuracies. Writing `N1` for the number of reference cases and `N0` for the number of non-cases, the quantity `tp * N0 + tn * N1` is an integer proportional to balanced accuracy, so combinations that genuinely tie are never separated by floating-point noise.

Value

An object of class `ptsdiag_plateau`: a list of two data frames.

- `summary`: one row per `delta`, with `delta`, `ba_best` (the best balanced accuracy), `ba_threshold` (the lowest balanced accuracy still on the plateau), `plateau_size` (number of combinations), `n_candidates` (size of this rule form's candidate space) and `plateau_proportion`.
- `sets`: one row per plateau member, with `delta`, `rank`, `combination_id`, `balanced_accuracy` and `ba_gap` (how far below the best it falls, in percentage points).

See Also

`score_all_combinations`, `bootstrap_stability`, `symptom_selection`.

Examples

```
# A compact 4-symptom search on a 120-row subset keeps the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:120, ],
                                 id_col = c("patient_id", "age", "sex"))
fit <- score_all_combinations(ptsd_data, n_symptoms = 4, n_required = 3,
                              show_progress = FALSE)

plateau <- compute_plateau(fit, delta = c(1, 5))
plateau$summary
head(plateau$sets)
```

`create_caps5_diagnosis` *Compute CAPS-5 DSM-5-TR PTSD diagnosis*

Description

Applies the DSM-5-TR PTSD diagnostic algorithm to CAPS-5 (Clinician-Administered PTSD Scale for DSM-5) item scores and returns a single-column dataframe indicating diagnostic status. Because CAPS-5 uses the same 20-item structure, 0–4 severity scale, and ≥ 2 symptom-presence threshold as the PCL-5, the diagnostic algorithm is identical.

Usage

```
create_caps5_diagnosis(data)
```

Arguments

`data`	A dataframe containing exactly 20 columns of CAPS-5 item severity scores (output of `rename_caps5_columns`). Columns must be named `symptom_1` through `symptom_20`, scored on a 0–4 scale, with no missing values.

Details

The DSM-5-TR diagnostic criteria applied are:

- Criterion B (Intrusion): ≥ 1 of items 1–5 with severity ≥ 2
- Criterion C (Avoidance): ≥ 1 of items 6–7 with severity ≥ 2
- Criterion D (Negative cognitions/mood): ≥ 2 of items 8–14 with severity ≥ 2
- Criterion E (Arousal/reactivity): ≥ 2 of items 15–20 with severity ≥ 2

All four criteria must be met for a positive diagnosis.

Unlike `create_icd11_diagnosis`, this function returns only the CAPS-5 diagnosis column (`PTSD_caps5`), not a PCL-5 reference column. This is because the CAPS-5 diagnosis is typically used as the gold-standard reference itself, not compared against a PCL-5 baseline.

The returned dataframe can be passed to `compare_diagnostic_systems` via its `caps5_data` parameter, or used directly for descriptive analyses.

Value

A `data.frame` with one logical column and one row per participant:

- `PTSD_caps5`: CAPS-5 DSM-5-TR diagnosis

Any carry-through columns present in `data` (e.g. an ID column added via `rename_caps5_columns`) are prepended in original order.

See Also

`rename_caps5_columns` for standardizing CAPS-5 column names.

`compare_diagnostic_systems` for comparing CAPS-5 against PCL-5 and optimized symptom combinations in a unified table.

`create_icd11_diagnosis` for the ICD-11 alternative criteria.

Examples

```
# Simulate CAPS-5 data (using same structure as PCL-5)
set.seed(42)
caps5_raw <- data.frame(matrix(sample(0:4, 400, replace = TRUE), ncol = 20))
caps5_data <- rename_caps5_columns(caps5_raw)
caps5_dx <- create_caps5_diagnosis(caps5_data)
head(caps5_dx)
table(caps5_dx$PTSD_caps5)
```

---

create_icd11_diagnosis

Apply ICD-11 PTSD diagnostic criteria to PCL-5 data

Description

Applies ICD-11 PTSD diagnostic criteria to PCL-5 item scores and returns a comparison dataframe against the full DSM-5-TR criteria. The output is directly compatible with `summarize_ptsd_changes` so that ICD-11 diagnostic accuracy can be computed on the same footing as optimized symptom combinations.

Usage

```
create_icd11_diagnosis(data)
```

Arguments

`data`	A dataframe containing exactly 20 columns of PCL-5 item scores (output of `rename_ptsd_columns`). Columns must be named `symptom_1` through `symptom_20`, scored on a 0–4 scale, with no missing values.

Details

ICD-11 PTSD requires ALL THREE of the following clusters to be met (symptom present = score ≥ 2 on original 0–4 scale):

1. **Re-experiencing in the present**: ≥ 1 of PCL-5 items 2, 3 (nightmares, flashbacks)
2. **Avoidance**: ≥ 1 of PCL-5 items 6, 7
3. **Sense of current threat**: ≥ 1 of PCL-5 items 17, 18 (hypervigilance, exaggerated startle)

A minimum of 3 symptoms total across all ICD-11 items (2, 3, 6, 7, 17, 18) must be present. This is automatically satisfied when all three cluster requirements are met but is enforced explicitly for clarity.

This is the *narrow* six-item mapping used across the published PCL-5-to-ICD-11 literature (e.g. Kuester et al. 2017, Schellong et al. 2019, Heeke et al. 2020, Pettrich et al. 2025): ICD-11 requires re-experiencing to have a here-and-now quality, which nightmares (item 2) and flashbacks (item 3) capture, while intrusive memories (item 1) as worded in the PCL-5 do not. To benchmark the broader seven-item variant (items 1, 2, 3, 6, 7, 17, 18) instead, compute it yourself and pass it to `compare_optimizations` as a custom fixed criterion:

```
broad <- rowSums(data[, paste0("symptom_", c(1, 2, 3))]  >= 2) >= 1 &
         rowSums(data[, paste0("symptom_", c(6, 7))]     >= 2) >= 1 &
         rowSums(data[, paste0("symptom_", c(17, 18))]   >= 2) >= 1
compare_optimizations(data, scenarios = list(
  "ICD-11 (broad)" = list(type = "fixed", criterion = broad,
                          symptoms = c(1, 2, 3, 6, 7, 17, 18))))
```

DSM-5-TR diagnosis (`PTSD_orig`) is computed using the same binarization logic as the rest of the package (`create_ptsd_diagnosis_binarized`).

Value

A `data.frame` with two logical columns and one row per participant:

- `PTSD_orig`: DSM-5-TR diagnosis (reference standard)
- `PTSD_icd11`: ICD-11 diagnosis

Any carry-through columns present in `data` (e.g. an ID column added via `rename_ptsd_columns`) are prepended in original order so results can be joined back to the source dataframe.

This dataframe can be passed directly to `summarize_ptsd_changes` or used as an input to `compare_diagnostic_systems`.

See Also

`compare_diagnostic_systems` for a unified cross-system comparison table.

`summarize_ptsd_changes` and `create_readable_summary` for computing and formatting diagnostic metrics.

Examples

```
# Apply ICD-11 criteria to the built-in simulated dataset
ptsd_data <- rename_ptsd_columns(simulated_ptsd,
                                 id_col = c("patient_id", "age", "sex"))
icd11_result <- create_icd11_diagnosis(ptsd_data)
head(icd11_result)

# Feed directly into the metrics pipeline
metrics <- summarize_ptsd_changes(icd11_result)
create_readable_summary(metrics)
```

`create_ptsd_diagnosis_binarized` *Determine PTSD diagnosis based on DSM-5 criteria using binarized scores*

Description

Determines whether DSM-5 diagnostic criteria for PTSD are met using binarized symptom scores (0/1) for PCL-5 items. This is an alternative to determine_ptsd_diagnosis() that works with pre-binarized data.

Usage

```
create_ptsd_diagnosis_binarized(data)
```

Arguments

`data`	A dataframe containing exactly 20 columns of PCL-5 item scores (output of rename_ptsd_columns) named symptom_1 to symptom_20. Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely Note: This function should only be used with raw symptom scores (output of rename_ptsd_columns) and not with data containing a total score column, as the internal binarization process would invalidate the total score.

Details

The function applies the DSM-5 diagnostic criteria for PTSD using binary indicators of symptom presence:

- Criterion B (Intrusion): At least 1 present symptom from items 1-5

- Criterion C (Avoidance): At least 1 present symptom from items 6-7
- Criterion D (Negative alterations in cognitions and mood): At least 2 present symptoms from items 8-14
- Criterion E (Alterations in arousal and reactivity): At least 2 present symptoms from items 15-20

Value

A dataframe with a single column "PTSD_orig" containing TRUE/FALSE values indicating whether DSM-5 diagnostic criteria are met based on binarized scores

Examples

```
# Create sample data
sample_data <- data.frame(
  matrix(sample(0:4, 20 * 10, replace = TRUE),
         nrow = 10,
         ncol = 20)
)
colnames(sample_data) <- paste0("symptom_", 1:20)

# Get diagnosis using binarized approach
diagnosis_results <- create_ptsd_diagnosis_binarized(sample_data)
diagnosis_results$PTSD_orig
```

---

create_ptsd_diagnosis_nonbinarized

Determine PTSD diagnosis based on DSM-5 criteria using non-binarized scores

---

Description

Determines whether DSM-5 diagnostic criteria for PTSD are met based on PCL-5 item scores, using the original non-binarized values (0-4 scale).

Usage

```
create_ptsd_diagnosis_nonbinarized(data)
```

Arguments

data	A dataframe that can be either: • Output of rename_ptsd_columns(): 20 columns named symptom_1 to symptom_20 • Output of calculate_ptsd_total(): 21 columns including symptom_1 to symptom_20 plus a 'total' column

Each symptom should be scored on a 0-4 scale where:

- 0 = Not at all
- 1 = A little bit
- 2 = Moderately
- 3 = Quite a bit
- 4 = Extremely

Details

The function applies the DSM-5 diagnostic criteria for PTSD:

- Criterion B (Intrusion): At least 1 symptom >= 2 from items 1-5
- Criterion C (Avoidance): At least 1 symptom >= 2 from items 6-7
- Criterion D (Negative alterations in cognitions and mood): At least 2 symptoms >= 2 from items 8-14
- Criterion E (Alterations in arousal and reactivity): At least 2 symptoms >= 2 from items 15-20

A symptom is considered present when rated 2 (Moderately) or higher.

Value

A dataframe with all original columns (including 'total' if present) plus an additional column "PTSD_orig" containing TRUE/FALSE values indicating whether DSM-5 diagnostic criteria are met

Examples

```
# Example with output from rename_ptsd_columns
sample_data1 <- data.frame(
  matrix(sample(0:4, 20 * 10, replace = TRUE),
         nrow = 10,
         ncol = 20)
)
colnames(sample_data1) <- paste0("symptom_", 1:20)
diagnosed_data1 <- create_ptsd_diagnosis_nonbinarized(sample_data1)

# Check diagnosis results
diagnosed_data1$PTSD_orig

# Example with output from calculate_ptsd_total
sample_data2 <- calculate_ptsd_total(sample_data1)
diagnosed_data2 <- create_ptsd_diagnosis_nonbinarized(sample_data2)

# Check diagnosis results
diagnosed_data2$PTSD_orig
```

`create_readable_summary` *Create readable summary of PTSD diagnostic changes*

Description

Formats the output of summarize_ptsd_changes() into a more readable table with proper labels and formatting of percentages and metrics.

Usage

```
create_readable_summary(summary_stats, DT = FALSE)
```

Arguments

`summary_stats`	A dataframe output from summarize_ptsd_changes() containing raw diagnostic metrics and counts
`DT`	Logical. If `TRUE`, return the summary as an interactive `datatable` widget. If `FALSE` (default), return a plain data.frame. The **DT** package must be installed when `DT = TRUE`.

Details

Reformats the diagnostic metrics into a presentation-ready format:

- Combines counts with percentages for diagnosed/non-diagnosed cases
- Rounds diagnostic accuracy metrics to 4 decimal places
- Provides clear column headers for all metrics

Value

A formatted data.frame (or a `datatable` widget when `DT = TRUE`) with the following columns:

- Scenario: Name of the diagnostic criterion
- Total Diagnosed: Count and percentage of diagnosed cases
- Total Non-Diagnosed: Count and percentage of non-diagnosed cases
- True Positive: Count of cases diagnosed under both criteria
- True Negative: Count of cases not diagnosed under either criterion
- Newly Diagnosed: Count of new positive diagnoses (false positive)
- Newly Non-Diagnosed: Count of new negative diagnoses (false negative)
- True Cases: Total correctly classified cases
- False Cases: Total misclassified cases
- Sensitivity, Specificity, PPV, NPV, Accuracy, Balanced Accuracy: Diagnostic accuracy metrics (4 decimals)

Examples

```
# Using the output from summarize_ptsd_changes
n_cases <- 100
sample_data <- data.frame(
  PTSD_orig = sample(c(TRUE, FALSE), n_cases, replace = TRUE),
  PTSD_alt1 = sample(c(TRUE, FALSE), n_cases, replace = TRUE)
)

# Generate and format summary
diagnostic_metrics <- summarize_ptsd_changes(sample_data)
readable_summary <- create_readable_summary(diagnostic_metrics)
print(readable_summary)
```

---

cross_validation *Perform k-fold cross-validation for PTSD diagnostic models*

Description

Validates PTSD diagnostic models using k-fold cross-validation to assess generalization performance and identify stable symptom combinations.

Usage

```
cross_validation(
  data,
  k = 5,
  score_by = "balanced_accuracy",
  seed = 123,
  n_symptoms = 6,
  n_required = 4,
  n_top = 3,
  DT = FALSE
)
```

Arguments

data	A dataframe containing the 20 PCL-5 item columns `symptom_1` through `symptom_20` (output of `rename_ptsd_columns`). Any additional non-symptom columns (e.g. an ID column passed via `rename_ptsd_columns(..., id_col = "patient_id")`) are carried through every fold and prepended to each `fold_results` entry so diagnoses can be joined back to the original dataframe.
k	Number of folds for cross-validation (default: 5)
score_by	Character string specifying optimization criterion: • "balanced_accuracy": Maximise balanced accuracy, the mean of sensitivity and specificity. Robust when one diagnostic class is much more common than the other. Default.

- "accuracy": Minimize total misclassifications (FP + FN, i.e. maximise overall accuracy).
- "sensitivity": Minimize false negatives only (i.e. maximise sensitivity relative to the full DSM-5-TR diagnosis).

`seed` Integer for random number generation reproducibility (default: 123)

`n_symptoms` Integer specifying how many symptoms per combination (default: 6). Must be between 1 and 20.

`n_required` Integer specifying how many symptoms must be present for diagnosis (default: 4). Must be between 1 and `n_symptoms`.

`n_top` Integer specifying how many top combinations to return (default: 3). Must be a positive integer.

`DT` Logical. If `TRUE`, return summaries as interactive `datatable` widgets. If `FALSE` (default), return plain data.frames. The **DT** package must be installed when `DT = TRUE`.

Details

The function:

1. Splits data into k stratified folds (preserving the proportion of diagnosed cases in each fold via `vfold_cv`)
2. For each fold, trains on k-1 folds and tests on the held-out fold
3. Identifies symptom combinations that appear across multiple folds
4. Calculates average performance metrics for repeated combinations

Two models are evaluated:

- Model without cluster representation: Any `n_required` of `n_symptoms` symptoms
- Model with cluster representation: `n_required` of `n_symptoms` symptoms with at least one from each cluster

If the **future.apply** package is installed and a `plan` has been set (e.g., `future::plan(future::multisession)`), folds are processed in parallel via `future_lapply`. On macOS (including Apple Silicon), use `future::multisession` rather than `future::multicore`, especially inside RStudio.

Value

A list containing:

- without_clusters: Results for model without cluster representation
  - fold_results: List of diagnostic comparisons for each fold
  - summary_by_fold: Detailed results for each fold (data.frame or DT widget)
  - combinations_summary: Average performance for combinations appearing in multiple folds (data.frame, DT widget, or NULL if no combinations repeat)
- with_clusters: Results for model with cluster representation
  - fold_results: List of diagnostic comparisons for each fold
  - summary_by_fold: Detailed results for each fold (data.frame or DT widget)
  - combinations_summary: Average performance for combinations appearing in multiple folds (data.frame, DT widget, or NULL if no combinations repeat)

Examples

```
# Use a 250-row subset of the bundled data to keep the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))


# 3-fold cross-validation of a compact 3-of-4 definition (a 4-symptom
# search keeps the example fast; use n_symptoms = 6, n_required = 4 for
# the classic rule)
cv_results <- cross_validation(ptsd_data, k = 3,
                               n_symptoms = 4, n_required = 3)

# View summary for each fold
cv_results$without_clusters$summary_by_fold

# View combinations that appeared multiple times
cv_results$without_clusters$combinations_summary
```

`diagnostic_metrics` *All diagnostic accuracy metrics for one or more two-by-two tables*

Description

Recomputes every metric from the counts, so a caller never has to trust a rounded sensitivity column or worry whether a stored metric was on the 0-1 or the 0-100 scale.

Usage

```
diagnostic_metrics(
  tp,
  fn,
  fp,
  tn,
  ci = c("wilson", "exact", "none"),
  conf_level = 0.95,
  ba_method = c("auto", "wilson", "exact", "wald")
)
```

Arguments

`tp`, `fn`, `fp`, `tn`	Counts of true positives, false negatives, false positives and true negatives. Vectorised and recycled together.
`ci`	`"wilson"` (default), `"exact"` for Clopper-Pearson, or `"none"`. Applies to the proportions **and**, through `ba_ci`, to balanced accuracy: asking for exact intervals gets exact intervals everywhere they are defined. Likelihood ratios always use `lr_ci`.

conf_level	Confidence level.
ba_method	Interval method for balanced accuracy: "auto" (default) follows ci, so ci = "exact" square-and-adds Clopper-Pearson components and ci = "wilson" square-and-adds Wilson ones. Set "wald" for the analytic normal-approximation variance, which is degenerate at a boundary – see ba_ci. With ci = "none" no balanced-accuracy interval is computed either.

Value

A data frame with one row per input. Proportions (sensitivity, specificity, ppv, npv, accuracy, ba, prevalence) are on the 0-1 scale; lr_pos and lr_neg are unbounded ratios, kappa can be negative, and pct_agreement is a percentage: the four counts, n, n_pos, n_neg, prevalence, then sensitivity, specificity, ppv, npv, accuracy and ba each with _lo and _hi, then lr_pos, lr_neg and kappa with intervals, and pct_agreement.

Examples

```
diagnostic_metrics(tp = 80, fn = 20, fp = 10, tn = 90)
diagnostic_metrics(tp = 2, fn = 4, fp = 1, tn = 176, ci = "exact")

# A perfect two-by-two still has an interval, because 183 observations are
# not infinite information.
diagnostic_metrics(tp = 6, fn = 0, fp = 0, tn = 177,
                   ci = "exact")[, c("ba", "ba_lo", "ba_hi")]
```

---

evaluate_definitions *Evaluate symptom definitions against a sample*

Description

Applies a set of pre-derived symptom definitions to a dataset and returns a performance table scoring each one against a reference standard. By default the reference is the sample's own full DSM-5-TR diagnosis; supply reference to score against an external standard instead (e.g. a clinician CAPS diagnosis). Because it needs only the definitions (symptom indices and rules) and a data frame, the same call can be run at a site that never saw the data the definitions were derived from.

Usage

```
evaluate_definitions(
  data,
  definitions,
  include_icd11 = TRUE,
  reference = NULL,
  include_full_pcl5 = NULL,
  tidy = FALSE,
  as_percent = FALSE
)
```

Arguments

data	A dataframe with the 20 PCL-5 item columns symptom_1 through symptom_20 (output of rename_ptsd_columns). Additional carry-through columns are ignored (but may be named by reference).
definitions	A named list of definitions, as returned by extract_definitions or as_definitions. Each element must contain symptoms (a list of integer vectors), n_required, and hierarchical (plus an optional clusters structure). A specification from read_combinations – or a list of them – is converted automatically via as_definitions.
include_icd11	Logical. If TRUE (default), append the ICD-11 criterion as a benchmark row.
reference	NULL (default) to score against the full DSM-5-TR PCL-5 diagnosis computed from data. Otherwise an external reference standard: a logical vector with one value per row, a 0/1-coded numeric vector, or a single string naming such a column in data. Missing values mark rows without a reference assessment; those rows are excluded (with a message).
include_full_pcl5	Logical or NULL (default). Whether to add a "Full 20-item PCL-5" ceiling row. NULL resolves to TRUE exactly when reference is external (where the row is informative) and FALSE otherwise (where it would be a self-comparison).
tidy	Logical. If FALSE (default), return the formatted display table (see create_readable_summary). If TRUE, return a plain analysis table matching summarize_top_combinations: one row per evaluated rule with Approach, Rank, Combination, the 2x2 counts, and numeric metrics – ready to filter, bind, or export without parsing labels.
as_percent	Logical. Only with tidy = TRUE: if TRUE, Sensitivity/Specificity/PPV/NPV/Accuracy/Balanced Accuracy are returned as percentages (0-100) instead of fractions (0-1). Default FALSE.

Details

Each definition is applied with its own rule via apply_symptom_combinations (using the definition's own cluster structure when present, otherwise the default PCL-5 clusters, when hierarchical = TRUE). When include_icd11 = TRUE, the ICD-11 criterion (create_icd11_diagnosis) is added as a fixed benchmark, computed locally on the supplied data.

With an external reference, rows with a missing reference value are excluded from the evaluation (with a message reporting how many), and a "Full 20-item PCL-5" ceiling row is added by default: the full DSM-5-TR PCL-5 diagnosis scored against the same reference. This row separates the cost of using a reduced symptom set from the intrinsic disagreement between the PCL-5 and the external standard – no reduced rule can be expected to exceed it.

Value

With tidy = FALSE, a formatted performance table (see create_readable_summary): one row for the reference standard (labelled PTSD_orig), one per definition (labelled by rule and symptom set), plus the optional "Full 20-item PCL-5" and "ICD-11" rows.

With tidy = TRUE, a data.frame with columns Approach, Rank (the combination's rank within its definition), Combination (comma-separated PCL-5 item numbers; NA for the full-PCL-5 ceiling row), TP, FN, FP, TN, Sensitivity, Specificity, PPV, NPV, Accuracy, Balanced Accuracy. The

reference self-comparison row is omitted. The layout matches summarize_top_combinations, so derivation and validation results can be combined with rbind().

See Also

extract_definitions, as_definitions, compare_optimizations, summarize_top_combinations.

Examples

```
# Use a 250-row subset and a small 4-symptom search to keep the example
# fast; omit `scenarios` to run the three default rules
ptsd <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                           id_col = c("patient_id", "age", "sex"))
comp <- compare_optimizations(
  ptsd,
  scenarios = list(
    "3/4 Non-hierarchical" = list(n_symptoms = 4, n_required = 3,
                                  hierarchical = FALSE)
  ),
  n_top = 10, show_progress = FALSE
)
definitions <- extract_definitions(comp, n = 3)

# Default: formatted table against the full DSM-5-TR PCL-5 diagnosis
evaluate_definitions(ptsd, definitions)

# Tidy analysis table (same layout as summarize_top_combinations())
evaluate_definitions(ptsd, definitions, tidy = TRUE)

# Against an external reference: the bundled general-population sample
# carries paired CAPS-5 items, standing in for a clinician diagnosis
gp   <- simulated_ptsd_genpop[1:400, ]
caps <- create_caps5_diagnosis(
  rename_caps5_columns(gp[, paste0("C", 1:20)])
)$PTSD_caps5
ptsd_gp <- rename_ptsd_columns(gp[, c("patient_id", paste0("S", 1:20))],
                               id_col = "patient_id")
ptsd_gp$caps <- caps
evaluate_definitions(ptsd_gp, definitions, reference = "caps", tidy = TRUE)
```

---

evaluate_sets *Evaluate given symptom combinations in one sample*

Description

Scores a supplied list of combinations against the full DSM-5-TR PCL-5 diagnosis, using the same collapsed-pattern matrix path as bootstrap_stability. Where score_all_combinations enumerates and scores an entire candidate space, this scores exactly the combinations it is given – which is what a validation site needs when it receives a plateau derived elsewhere.

Usage

```
evaluate_sets(data, sets, n_required, clusters = NULL, chunk_size = 1000L)
```

Arguments

data	A data frame with symptom_1 ... symptom_20.
sets	A combination specification, see as_set_matrix.
n_required	Endorsements required for a positive decision.
clusters	NULL for the non-hierarchical rule, or a named list of integer vectors to additionally require an endorsed symptom in every cluster.
chunk_size	Combinations per matrix block. Affects memory, not results.

Details

Respondents are collapsed to unique binarized response patterns first, so cost scales with the number of distinct patterns rather than the sample size.

The result carries one row per combination. That is the right granularity for local analysis and the wrong granularity to share: per-combination two-by-two counts across a large candidate space are invertible back towards the response-pattern distribution. Use transport_plateau to produce something releasable.

Value

A data frame with one row per combination: set_id, tp, fn, fp, tn, n, sensitivity, specificity, ba.

See Also

transport_plateau for the shareable summary, score_all_combinations for an exhaustive search.

Examples

```
data(simulated_ptsd_genpop)
prepared <- rename_ptsd_columns(
  simulated_ptsd_genpop[, c("patient_id", paste0("S", 1:20))],
  id_col = "patient_id")
evaluate_sets(prepared, c("2_3_6_7_17_18", "1_2_3_4_5_6"), n_required = 4)
```

exact_ci *Clopper-Pearson (exact) interval for a binomial proportion*

Description

Required wherever a denominator is small enough that the normal approximation misbehaves – a clinician-interview subset, for instance. Slower than `wilson_ci` because it calls [stats::binom.test()] per row.

Usage

```
exact_ci(x, n, conf_level = 0.95)
```

Arguments

`x`	Number of successes. Vectorised.
`n`	Number of trials. Vectorised.
`conf_level`	Confidence level.

Value

A two-column matrix with columns `lo` and `hi`, on the 0-1 scale.

Examples

```
exact_ci(2, 6)
```

extract_definitions *Extract portable symptom definitions from a comparison*

Description

Pulls the top symptom combinations of each optimized scenario out of a `compare_optimizations` result and returns them as a compact, shareable object. Each definition is described only by its symptom indices and the rule needed to apply it (how many must be present, and whether cluster representation is required), so the object contains no participant-level data and can be shared across sites.

Usage

```
extract_definitions(comparison, n = 5)
```

Arguments

`comparison`	A `ptsdiag_comparison` object from `compare_optimizations`.
`n`	Integer. Number of top combinations to keep per optimized scenario (default 5). Capped at the number available.

Details

For each `type = "optimize"` scenario in the comparison, the rule (`n_required`, `hierarchical`) is read from `comparison$config`, so the only thing the user supplies is how many combinations to carry per scenario. Fixed scenarios (e.g. ICD-11) are skipped, because their symptom set is published rather than derived.

The result pairs with `evaluate_definitions`: extract the definitions from one sample, then evaluate them in any sample.

Value

A named list (one element per optimized scenario). Each element is a list with:

- `symptoms`: list of integer vectors (the top-n combinations).
- `n_required`: integer threshold for that scenario.
- `hierarchical`: logical, whether cluster representation is required.

See Also

`evaluate_definitions`, `as_definitions` for building the same object from combinations imported with `read_combinations`, `compare_optimizations`.

Examples

```
# Use a 250-row subset and a small 4-symptom search to keep the example
# fast; omit `scenarios` to run the three default rules
ptsd <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                            id_col = c("patient_id", "age", "sex"))
comp <- compare_optimizations(
  ptsd,
  scenarios = list(
    "3/4 Non-hierarchical" = list(n_symptoms = 4, n_required = 3,
                                  hierarchical = FALSE)
  ),
  n_top = 10, show_progress = FALSE
)
definitions <- extract_definitions(comp, n = 5)
lapply(definitions, function(d) d$symptoms)
```

`five_number` *Five-number summary of a numeric vector, as a one-row data frame*

Description

The shape of a multi-site return. When a plateau of near-optimal combinations is evaluated at a site that cannot release per-combination results, these six numbers are what travels: the minimum and maximum give the whisker, the quartiles the box, the median the point. The combination-to-value linkage is discarded, which is what makes the summary safe to share – see `transport_plateau`.

Usage

```
five_number(x)
```

Arguments

x	A numeric vector. Missing values are dropped.

Value

A one-row data frame: n_sets, min, q1, median, q3, max. An empty input gives n_sets = 0 and NA elsewhere.

Examples

```
five_number(c(0.81, 0.83, 0.84, 0.85, 0.88))
```

---

fmt_est_ci *Format an estimate and its interval for a table*

Description

Format an estimate and its interval for a table

Usage

```
fmt_est_ci(est, lo, hi, digits = 1, scale = 100)
```

Arguments

est, lo, hi	Estimate and interval bounds, on the 0-1 scale by default.
digits	Decimal places.
scale	Multiplier applied before formatting; 100 turns proportions into percentages.

Value

A character vector such as "88.3 (87.4–89.1)" (the bounds are separated by an en dash, not a hyphen), or an em dash where the estimate is missing.

Examples

```
m <- diagnostic_metrics(tp = 80, fn = 20, fp = 10, tn = 90)
fmt_est_ci(m$sensitivity, m$sensitivity_lo, m$sensitivity_hi)
```

fmt_ratio_ci *Format a ratio and its interval for a table*

Description

Format a ratio and its interval for a table

Usage

```
fmt_ratio_ci(est, lo, hi, digits = 2)
```

Arguments

est, lo, hi	Estimate and interval bounds.
digits	Decimal places.

Value

A character vector such as "5.21 (4.80–5.66)" (the bounds are separated by an en dash, not a hyphen), or an em dash where a zero cell leaves the ratio undefined.

Examples

```
m <- diagnostic_metrics(tp = 80, fn = 20, fp = 10, tn = 90)
fmt_ratio_ci(m$lr_pos, m$lr_pos_lo, m$lr_pos_hi)
```

holdout_validation *Perform holdout validation for PTSD diagnostic models*

Description

Validates PTSD diagnostic models using a train-test split approach (holdout validation). Trains the model on a portion of the data and evaluates performance on the held-out test set.

Usage

```
holdout_validation(
  data,
  train_ratio = 0.7,
  score_by = "balanced_accuracy",
  seed = 123,
  n_symptoms = 6,
  n_required = 4,
  n_top = 3,
  DT = FALSE
)
```

Arguments

data	A dataframe containing the 20 PCL-5 item columns symptom_1 through symptom_20 (output of rename_ptsd_columns). Each symptom should be scored on a 0-4 scale. Any additional non-symptom columns (e.g. an ID column passed via rename_ptsd_columns(..., id_col = "patient_id")) are carried through the train/test split and prepended to test_results so diagnoses can be joined back to the original dataframe.
train_ratio	Numeric between 0 and 1 indicating proportion of data for training (default: 0.7 for 70/30 split)
score_by	Character string specifying optimization criterion: • "balanced_accuracy": Maximise balanced accuracy, the mean of sensitivity and specificity. Robust when one diagnostic class is much more common than the other. Default. • "accuracy": Minimize total misclassifications (FP + FN, i.e. maximise overall accuracy). • "sensitivity": Minimize false negatives only (i.e. maximise sensitivity relative to the full DSM-5-TR diagnosis).
seed	Integer for random number generation reproducibility (default: 123)
n_symptoms	Integer specifying how many symptoms per combination (default: 6). Must be between 1 and 20.
n_required	Integer specifying how many symptoms must be present for diagnosis (default: 4). Must be between 1 and n_symptoms.
n_top	Integer specifying how many top combinations to return (default: 3). Must be a positive integer.
DT	Logical. If TRUE, return summaries as interactive datatable widgets. If FALSE (default), return plain data.frames. The **DT** package must be installed when DT = TRUE.

Details

The function:

1. Splits data into training and test sets based on train_ratio
2. Finds optimal symptom combinations on training data
3. Evaluates these combinations on test data
4. Compares results to original DSM-5 diagnoses

Two models are evaluated:

- Model without cluster representation: Any n_required of n_symptoms symptoms
- Model with cluster representation: n_required of n_symptoms symptoms with at least one from each cluster

Value

A list containing:

- without_clusters: Results for model without cluster representation
  - best_combinations: The `n_top` best symptom combinations from training
  - test_results: Diagnostic comparison on test data
  - summary: Formatted summary statistics (data.frame or DT widget)
- with_clusters: Results for model with cluster representation
  - best_combinations: The `n_top` best symptom combinations from training
  - test_results: Diagnostic comparison on test data
  - summary: Formatted summary statistics (data.frame or DT widget)

Examples

```
# Use a 250-row subset of the bundled data to keep the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))


# Validate a compact 3-of-5 definition (a 5-symptom search keeps the
# example fast; use n_symptoms = 6, n_required = 4 for the classic rule)
validation_results <- holdout_validation(ptsd_data, train_ratio = 0.7,
                                         n_symptoms = 5, n_required = 3)

# Access results
validation_results$without_clusters$summary
validation_results$with_clusters$summary
```

---

icd11_items *The ICD-11 PTSD criterion as PCL-5 items*

Description

The mapping `create_icd11_diagnosis` scores: two items per ICD-11 cluster, at least one endorsed in each. Exposed so that a table describing the operationalisation is generated from the same definition that is scored, and cannot drift from it.

Usage

```
icd11_items(flat = TRUE)
```

Arguments

`flat`	If `TRUE` (default) return a sorted integer vector of the six items; if `FALSE` return the named list of clusters.

Details

Note that PCL-5 item 1 is deliberately absent. Analyses published before PTSDdiag 0.4.0 used a seven-item mapping that included it.

Value

An integer vector of length 6, or a named list of three integer pairs.

Examples

```
icd11_items()
icd11_items(flat = FALSE)
```

---

icd11_performance *ICD-11 performance in one sample*

Description

The two-by-two table of `create_icd11_diagnosis` against the full DSM-5-TR PCL-5 diagnosis, in the same shape `evaluate_sets` returns, so the ICD-11 comparator can be bound to a table of derived combinations.

Usage

```
icd11_performance(data)
```

Arguments

`data`	A data frame with `symptom_1` ... `symptom_20`.

Value

A one-row data frame: `set_id` (the six ICD-11 items), `tp`, `fn`, `fp`, `tn`.

Examples

```
data(simulated_ptsd_genpop)
prepared <- rename_ptsd_columns(
  simulated_ptsd_genpop[, c("patient_id", paste0("S", 1:20))],
  id_col = "patient_id")
icd11_performance(prepared)
```

kappa_ci — *Cohen's kappa for a two-by-two table, with a large-sample interval*

Description

Uses `SE = sqrt(po (1 - po) / (n (1 - pe)^2))`. With a small denominator the interval is wide, which is the honest answer rather than a defect.

Usage

```
kappa_ci(tp, fn, fp, tn, conf_level = 0.95)
```

Arguments

`tp, fn, fp, tn`	Counts of true positives, false negatives, false positives and true negatives. Vectorised and recycled together.
`conf_level`	Confidence level.

Value

A three-column matrix with columns `est`, `lo` and `hi`.

Examples

```
kappa_ci(tp = 80, fn = 20, fp = 10, tn = 90)
```

lr_ci — *Likelihood ratios with log-normal (Simel) intervals*

Description

Likelihood ratios with log-normal (Simel) intervals

Usage

```
lr_ci(tp, fn, fp, tn, which = c("pos", "neg"), conf_level = 0.95)
```

Arguments

`tp, fn, fp, tn`	Counts of true positives, false negatives, false positives and true negatives. Vectorised and recycled together.
`which`	`"pos"` for the positive likelihood ratio, `"neg"` for the negative one.
`conf_level`	Confidence level.

Value

A three-column matrix with columns `est`, `lo` and `hi`. Entries are infinite or `NaN` where a zero cell makes the ratio undefined.

Examples

```
lr_ci(tp = 80, fn = 20, fp = 10, tn = 90, which = "pos")
```

`n_candidates` *Size of a candidate space*

Description

How many symptom combinations a search will evaluate. With a cluster structure only combinations covering every cluster are candidates, which for six PCL-5 items out of twenty is 13,685 rather than 38,760 – the denominator a plateau proportion has to be read against.

Usage

```
n_candidates(n_symptoms = 6L, clusters = NULL, n_total = 20L)
```

Arguments

`n_symptoms`	Combination size.
`clusters`	`NULL`, or a named list of integer vectors.
`n_total`	Number of items to choose from.

Value

A single integer.

Examples

```
# Small spaces make the point in milliseconds; the six-symptom equivalents
# are 38,760 and 13,685. A cluster-constrained space needs at least one item
# per cluster, so four is the smallest that exists here.
n_candidates(3)
n_candidates(4, clusters = list(B = 1:5, C = 6:7, D = 8:14, E = 15:20))
```

`optimize_combinations` *Find optimal symptom combinations for diagnosis (non-hierarchical)*

Description

Identifies the best symptom combinations for PTSD diagnosis where a specified number of symptoms must be present, regardless of their cluster membership. This is a generalized version that allows configuring the number of symptoms per combination, the required threshold, and how many top results to return.

Usage

```
optimize_combinations(
  data,
  n_symptoms = 6,
  n_required = 4,
  n_top = 3,
  score_by = "balanced_accuracy",
  DT = FALSE,
  show_progress = TRUE
)
```

Arguments

`data`	A dataframe containing exactly 20 columns with PCL-5 item scores (output of `rename_ptsd_columns`). Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely
`n_symptoms`	Integer specifying how many symptoms per combination (default: 6). Must be between 1 and 20.
`n_required`	Integer specifying how many symptoms must be present for diagnosis (default: 4). Must be between 1 and `n_symptoms`.
`n_top`	Integer specifying how many top combinations to return (default: 3). Must be a positive integer.
`score_by`	Character string specifying optimization criterion: • "balanced_accuracy": Maximise balanced accuracy, the mean of sensitivity and specificity. Robust when one diagnostic class is much more common than the other. Default. • "accuracy": Minimize total misclassifications (FP + FN, i.e. maximise overall accuracy).

- "sensitivity": Minimize false negatives only (i.e. maximise sensitivity relative to the full DSM-5-TR diagnosis).

`DT` Logical. If `TRUE`, return the summary as an interactive `datatable` widget. If `FALSE` (default), return a plain data.frame. The **DT** package must be installed when `DT = TRUE`.

`show_progress` Logical. If `TRUE` (default), display a progress bar while evaluating combinations. Set to `FALSE` for batch or non-interactive use.

Details

The function:

1. Tests all possible combinations of `n_symptoms` symptoms from the 20 PCL-5 items
2. Requires `n_required` symptoms to be present (>=2 on original 0-4 scale) for diagnosis
3. Identifies the `n_top` combinations that best match the original DSM-5 diagnosis

Optimization can be based on:

- Maximizing balanced accuracy, the mean of sensitivity and specificity (the default)
- Minimizing false cases (both false positives and false negatives)
- Minimizing only false negatives (newly non-diagnosed cases)

The symptom clusters in PCL-5 are:

- Items 1-5: Intrusion symptoms (Criterion B)
- Items 6-7: Avoidance symptoms (Criterion C)
- Items 8-14: Negative alterations in cognitions and mood (Criterion D)
- Items 15-20: Alterations in arousal and reactivity (Criterion E)

Value

A list containing:

- best_symptoms: List of `n_top` vectors, each containing `n_symptoms` symptom numbers representing the best combinations found
- diagnosis_comparison: Dataframe comparing original DSM-5 diagnosis with diagnoses based on the best combinations. If `data` carried non-symptom columns (e.g. an ID column added via `rename_ptsd_columns`), those are prepended in original order.
- summary: Diagnostic accuracy metrics for each combination. A data.frame by default, or an interactive `datatable` if `DT = TRUE`.
- n_tied: Integer. Number of additional combinations that scored identically to the best combination but are not included in the top results. When `n_tied > 0`, the reported "best" combination is one of several equivalent solutions. Ties are broken by lexicographic order of symptom indices.

Examples

```
# Use a 250-row subset of the bundled data to keep the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))


# Find best 6-symptom combinations requiring 4 present (classic defaults,
# optimized for balanced accuracy)
results <- optimize_combinations(ptsd_data, n_symptoms = 6, n_required = 4,
             score_by = "balanced_accuracy")

# Find best 5-symptom combinations requiring 3 present, return top 5,
# this time minimizing total misclassifications
results2 <- optimize_combinations(ptsd_data, n_symptoms = 5, n_required = 3,
              n_top = 5, score_by = "accuracy")

# Get symptom numbers
results$best_symptoms

# Check how many combinations tied with the best
results$n_tied

# View summary statistics
results$summary
```

---

optimize_combinations_clusters

Find optimal symptom combinations for diagnosis (hierarchical/cluster-based)

Description

Identifies the best symptom combinations for PTSD diagnosis where a specified number of symptoms must be present and must include at least one symptom from each defined cluster. This maintains the hierarchical structure of the diagnostic criteria while allowing configurable parameters.

Usage

```
optimize_combinations_clusters(
  data,
  n_symptoms = 6,
  n_required = 4,
  n_top = 3,
  score_by = "balanced_accuracy",
  clusters,
  DT = FALSE,
  show_progress = TRUE
)
```

Arguments

`data`	A dataframe containing exactly 20 columns with PCL-5 item scores (output of `rename_ptsd_columns`). Each symptom should be scored on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely
`n_symptoms`	Integer specifying how many symptoms per combination (default: 6). Must be at least as large as the number of clusters.
`n_required`	Integer specifying how many symptoms must be present for diagnosis (default: 4). Must be between 1 and `n_symptoms`.
`n_top`	Integer specifying how many top combinations to return (default: 3). Must be a positive integer.
`score_by`	Character string specifying optimization criterion: • "balanced_accuracy": Maximise balanced accuracy, the mean of sensitivity and specificity. Robust when one diagnostic class is much more common than the other. Default. • "accuracy": Minimize total misclassifications (FP + FN, i.e. maximise overall accuracy). • "sensitivity": Minimize false negatives only (i.e. maximise sensitivity relative to the full DSM-5-TR diagnosis).
`clusters`	A named list of integer vectors defining the cluster structure. Each list element represents one cluster, with the integer vector specifying which symptom indices belong to that cluster. Cluster elements must not overlap. This parameter is required (no default). For PCL-5: `list(B = 1:5, C = 6:7, D = 8:14, E = 15:20)`
`DT`	Logical. If `TRUE`, return the summary as an interactive `datatable` widget. If `FALSE` (default), return a plain data.frame. The **DT** package must be installed when `DT = TRUE`.
`show_progress`	Logical. If `TRUE` (default), display a progress bar while evaluating combinations. Set to `FALSE` for batch or non-interactive use.

Details

The function:

1. Generates valid combinations ensuring representation from all clusters
2. Requires `n_required` symptoms to be present (>=2 on original 0-4 scale) for diagnosis
3. Validates that present symptoms include at least one from each cluster
4. Identifies the `n_top` combinations that best match the original DSM-5 diagnosis

The `clusters` parameter must be a named list specifying the cluster structure. For PCL-5, the standard clusters are:

- Cluster B (Intrusion): Items 1-5
- Cluster C (Avoidance): Items 6-7
- Cluster D (Negative alterations in cognitions and mood): Items 8-14
- Cluster E (Alterations in arousal and reactivity): Items 15-20

Optimization can be based on:

- Maximizing balanced accuracy, the mean of sensitivity and specificity (the default)
- Minimizing false cases (both false positives and false negatives)
- Minimizing only false negatives (newly non-diagnosed cases)

Value

A list containing:

- best_symptoms: List of `n_top` vectors, each containing `n_symptoms` symptom numbers representing the best combinations found
- diagnosis_comparison: Dataframe comparing original DSM-5 diagnosis with diagnoses based on the best combinations. If `data` carried non-symptom columns (e.g. an ID column added via `rename_ptsd_columns`), those are prepended in original order.
- summary: Diagnostic accuracy metrics for each combination. A data.frame by default, or an interactive `datatable` if `DT = TRUE`.

Examples

```
# Use a 250-row subset of the bundled data to keep the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))


# Find best hierarchical combinations with PCL-5 clusters (a 5-symptom
# search keeps the example fast; the classic rule uses n_symptoms = 6,
# n_required = 4)
pcl5_clusters <- list(B = 1:5, C = 6:7, D = 8:14, E = 15:20)
results <- optimize_combinations_clusters(ptsd_data, n_symptoms = 5,
             n_required = 3, score_by = "balanced_accuracy",
             clusters = pcl5_clusters)

# Get symptom numbers
results$best_symptoms

# View summary statistics
results$summary
```

pcl5_item_labels *PCL-5 item labels, in DSM-5-TR order*

Description

The twenty short item labels used across this package's plots and tables. Position matters: `pcl5_item_labels()[6]` is the label of `symptom_6`.

Usage

```
pcl5_item_labels()
```

Value

A character vector of length 20.

Examples

```
pcl5_item_labels()[c(2, 3, 6, 7, 17, 18)]
```

plot_symptom_frequency *Heatmap of PCL-5 symptom selection frequency across optimization scenarios*

Description

Visualises how often each of the 20 PCL-5 symptoms is selected across the top combinations of each optimization scenario in a `compare_optimizations` result. Replicates the symptom-frequency heatmap (Figure 1) of the PTSDdiag preprint and helps identify "core" symptoms that recur across data-driven combinations.

Usage

```
plot_symptom_frequency(
  comparison,
  type = c("relative", "absolute", "enrichment"),
  show_overall = TRUE,
  overall_includes_fixed = FALSE,
  symptom_labels = NULL,
  clusters = NULL,
  low_colour = "#f7fbff",
  high_colour = "#084594"
)
```

Arguments

comparison	A ptsdiag_comparison object.
type	"relative" (default; fill = RelFreq, percentage labels), "absolute" (fill = Count), or "enrichment" (fill = RelFreq / Baseline, diverging around 1).
show_overall	Logical. Include the pooled OVERALL row. Default TRUE.
overall_includes_fixed	Logical. If TRUE, fixed criteria contribute to the OVERALL row. Default FALSE.
symptom_labels	Optional character vector of length 20 used to label the x-axis ticks. Default uses the numeric indices 1:20.
clusters	Named list of item indices defining the clusters used by the hierarchical scenarios, for the chance baseline. Defaults to the standard DSM-5-TR PCL-5 clusters.
low_colour, high_colour	Gradient endpoints for the fill scale. For type = "enrichment" they anchor the below-chance and above-chance ends of a diverging scale.

Details

Each tile shows the frequency with which a symptom appears in the stored combinations of a scenario. Fixed criteria (e.g. ICD-11) appear as rows with cells at RelFreq = 1 on their included symptoms and RelFreq = 0 elsewhere. The optional OVERALL row pools across optimization scenarios by default (set overall_includes_fixed = TRUE to include fixed criteria in the pool). It is rendered in a separate facet so it is visually distinct from the per-scenario rows.

Selection frequencies are zero-sum across a fixed number of combinations, so type = "enrichment" is usually the more informative view: it fills each tile with the ratio of the observed frequency to the frequency expected by chance for that scenario's candidate space, on a scale diverging around 1. This matters most for hierarchical scenarios, where the chance baseline differs between clusters. See symptom_frequency for the underlying columns.

Requires the **ggplot2** package.

Value

A ggplot object. Users can extend it with additional layers, themes, or labels via the usual + operator.

See Also

compare_optimizations, symptom_frequency, summarize_top_combinations.

Examples

```
# Use a 250-row subset and a small 4-symptom search to keep the example
# fast; omit `scenarios` to run the three default rules
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))
comp <- compare_optimizations(
  ptsd_data,
  scenarios = list(
```

```
    "3/4 Non-hierarchical" = list(n_symptoms = 4, n_required = 3,
                                  hierarchical = FALSE)
  ),
  include_icd11 = TRUE, n_top = 5, show_progress = FALSE
)
plot_symptom_frequency(comp)
```

`plot_symptom_selection` *Plot symptom selection against the chance baseline*

Description

Draws the per-item selection frequencies from `symptom_selection` together with the frequency expected by chance. The baseline is drawn item by item rather than as a single reference line, because under the hierarchical rule it differs by cluster: a flat line would make two-item clusters look preferred when they are simply required.

Usage

```
plot_symptom_selection(
  selection,
  type = c("frequency", "enrichment"),
  weighted = FALSE,
  symptom_labels = NULL,
  bar_colour = "#4292c6",
  baseline_colour = "#252525"
)
```

Arguments

`selection`	A `symptom_selection` result.
`type`	Character. `"frequency"` (default) draws the selection frequency of each item with its own chance baseline marked; `"enrichment"` draws the ratio of the two, with a reference line at 1.
`weighted`	Logical. If `TRUE`, use the stability weighted columns (`pi_freq`, `pi_enrichment`), which requires `symptom_selection` to have been given a `bootstrap_stability` result. Default `FALSE`.
`symptom_labels`	Optional character vector of length 20 for the axis. Defaults to the `label` column of `selection`, or item numbers.
`bar_colour`	Fill colour used when the items have no cluster labels.
`baseline_colour`	Colour of the chance-baseline markers.

Value

A ggplot object.

See Also

symptom_selection, plot_symptom_frequency.

Examples

```
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:120, ],
                                 id_col = c("patient_id", "age", "sex"))
fit <- score_all_combinations(ptsd_data, n_symptoms = 4, n_required = 3,
                              show_progress = FALSE)
selection <- symptom_selection(fit)
plot_symptom_selection(selection)
```

---

print.ptsdiag_comparison

Print method for ptsdiag_comparison objects

Description

Print method for ptsdiag_comparison objects

Usage

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

Arguments

x	A ptsdiag_comparison object.
...	Unused.

Value

The input object invisibly.

print.ptsdiag_plateau *Print method for ptsdiag_plateau objects*

Description

Print method for ptsdiag_plateau objects

Usage

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

Arguments

x	A ptsdiag_plateau object.
...	Ignored.

Value

x, invisibly.

print.ptsdiag_rule_forms *Print method for ptsdiag_rule_forms objects*

Description

Print method for ptsdiag_rule_forms objects

Usage

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

Arguments

x	A ptsdiag_rule_forms object.
...	Ignored.

Value

x, invisibly.

`print.ptsdiag_stability` *Print method for ptsdiag_stability objects*

Description

Print method for ptsdiag_stability objects

Usage

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

Arguments

`x`	A `ptsdiag_stability` object.
`...`	Ignored.

Value

`x`, invisibly.

`print.ptsdiag_transport` *Print method for ptsdiag_transport objects*

Description

Print method for ptsdiag_transport objects

Usage

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

Arguments

`x`	A `ptsdiag_transport` object.
`...`	Ignored.

Value

`x`, invisibly.

`read_combinations` *Read symptom combinations from a JSON file*

Description

Imports symptom combinations from a JSON file previously created by `write_combinations`. The returned list contains all fields needed to apply the combinations to new data via `apply_symptom_combinations`.

Usage

```
read_combinations(file)
```

Arguments

`file`	Character string. Path to the JSON file to read.

Details

The function validates the imported data using the same checks as `apply_symptom_combinations`, ensuring that the file contains valid combinations, a valid `n_required` threshold, and (if present) a valid cluster structure.

If the file was created with a different version of PTSDdiag than the one currently installed, an informational message is displayed.

Value

A named list with the following elements:

combinations List of numeric vectors. Each vector contains symptom indices for one combination.

combination_ids Character vector of canonical combination IDs (sorted symptom indices joined by underscores, e.g. `"4_6_7_17_19_20"`). Computed from the combinations if not present in the file (backward compatibility with files created before v0.2.1).

ranks Integer vector of ranks (1 = best). Computed from list position if not present in the file.

n_required Numeric. Number of symptoms required for a positive diagnosis.

clusters `NULL` for non-hierarchical combinations, or a named list of numeric vectors defining the cluster structure.

parameters Named list with additional metadata: `n_symptoms` and `score_by` (may be `NULL` if not recorded).

description Character string with the user-provided description.

label Character string with the display label stored by `write_combinations`, or `NA` for files written without one (including files from versions before 0.4.0). Used by `as_definitions` to name the definition.

ptsddiag_version Character string indicating which package version created the file.

created_at Character string with the creation timestamp.

The returned list carries class `ptsdiag_spec`, so it can be handed directly to `as_definitions` or `evaluate_definitions` (which converts it automatically). For row-level diagnoses, the `combinations`, `n_required`, and `clusters` elements can also be passed to `apply_symptom_combinations`:

```
spec <- read_combinations("my_combos.json")
result <- apply_symptom_combinations(
  data, spec$combinations, spec$n_required, spec$clusters
)
```

See Also

`write_combinations` to export combinations to a JSON file.

`as_definitions` to convert one or more imported specifications into the definitions list used by `evaluate_definitions`.

`apply_symptom_combinations` to apply imported combinations to new data.

Examples

```
# Write example combinations
my_combos <- list(
  c(1, 6, 8, 10, 15, 19),
  c(2, 7, 9, 11, 16, 20)
)
tmp <- tempfile(fileext = ".json")
write_combinations(my_combos, tmp, n_required = 4,
                   score_by = "balanced_accuracy")

# Read them back
spec <- read_combinations(tmp)
spec$combinations
spec$n_required

# Apply to data (example workflow)
# comparison <- apply_symptom_combinations(
#   new_data, spec$combinations, spec$n_required, spec$clusters
# )

# Clean up
unlink(tmp)
```

---

read_transport *Read a return bundle*

Description

The inverse of `write_transport`. Optional parts that a site did not produce come back as `NULL` rather than erroring.

Usage

```
read_transport(dir, sample_id = NULL)
```

Arguments

dir	Directory holding the bundle.
sample_id	Sample identifier, which is also the filename prefix. If NULL, it is inferred from the single *_transport_summary.csv present, and an error is raised if there is more than one.

Value

A list of six elements:

- sample_id: the identifier the bundle was written under, whether it was supplied or inferred from the filenames.
- summary, named: the two tables described in transport_plateau.
- caps_agreement, sample_info, flow: the optional parts, or NULL where the site did not produce them.

See Also

write_transport, transport_plateau.

Examples

```
ptsd_data <- rename_ptsd_columns(
  simulated_ptsd_genpop[1:400, c("patient_id", paste0("S", 1:20))],
  id_col = "patient_id")
fit <- score_all_combinations(ptsd_data, n_symptoms = 3, n_required = 2,
                              show_progress = FALSE)
plateau <- compute_plateau(fit, delta = 1)
bundle <- transport_plateau(
  ptsd_data,
  plateau_specs = list(list(derivation = "example", rule_form = "flat_2_3",
                            delta = 1, n_required = 2, clusters = NULL,
                            sets = plateau$sets$combination_id)),
  named_specs = list(list(derivation = "example", rule_form = "flat_2_3",
                          n_required = 2, clusters = NULL,
                          sets = fit$combination_id[1:3], ranks = 1:3)),
  sample_id = "site_a"
)

out <- file.path(tempdir(), "transport_roundtrip")
dir.create(out, showWarnings = FALSE)
write_transport(bundle, out)

returned <- read_transport(out)
returned$sample_id
returned$summary
```

```
unlink(out, recursive = TRUE)
```

`rename_caps5_columns` *Rename CAPS-5 symptom columns*

Description

Standardizes column names in CAPS-5 (Clinician-Administered PTSD Scale for DSM-5) data by renaming them to a consistent format (`symptom_1` through `symptom_20`). This standardization allows CAPS-5 data to be used with the same downstream functions as PCL-5 data (e.g., `create_caps5_diagnosis`, `apply_symptom_combinations`).

Usage

```
rename_caps5_columns(data, id_col = NULL)
```

Arguments

`data`	A dataframe containing exactly 20 columns of CAPS-5 item severity ratings (plus any columns named in `id_col` for carry-through). Scores are on a 0–4 scale where: • 0 = Absent • 1 = Mild / subthreshold • 2 = Moderate / threshold (counts toward diagnosis) • 3 = Severe / markedly elevated • 4 = Extreme / incapacitating
`id_col`	Optional character vector naming column(s) in `data` to preserve as identifiers. These columns propagate through the workflow and can be used to merge per-row diagnoses back to the original dataframe. Defaults to `NULL`.

Details

The function assumes the input data contains exactly 20 columns corresponding to the 20 CAPS-5 items. Each item is a single severity rating (0–4) assigned by the clinician, who combines information about frequency and intensity into that score. The columns are renamed sequentially from `symptom_1` to `symptom_20`, maintaining their original order.

The CAPS-5 items map to the same DSM-5 PTSD symptom clusters as the PCL-5:

- symptom_1 to symptom_5: Intrusion symptoms (Criterion B)
- symptom_6 to symptom_7: Avoidance symptoms (Criterion C)
- symptom_8 to symptom_14: Negative alterations in cognitions and mood (Criterion D)
- symptom_15 to symptom_20: Alterations in arousal and reactivity (Criterion E)

The output naming (`symptom_1:symptom_20`) is intentionally identical to the PCL-5 convention so that downstream functions such as `apply_symptom_combinations` and `compare_diagnostic_systems` work transparently on CAPS-5 data.

Value

A dataframe with CAPS-5 columns renamed to `symptom_1` through `symptom_20`. If `id_col` is supplied, the named columns are prepended (in original order).

See Also

`rename_ptsd_columns` for the PCL-5 equivalent.

`create_caps5_diagnosis` for computing a CAPS-5 DSM-5 diagnosis from the renamed data.

Examples

```
# Example with simulated CAPS-5 data
caps5_data <- data.frame(
  matrix(sample(0:4, 20 * 10, replace = TRUE),
         nrow = 10,
         ncol = 20)
)
renamed_caps5 <- rename_caps5_columns(caps5_data)
colnames(renamed_caps5)  # symptom_1 through symptom_20
```

`rename_ptsd_columns` *Rename PTSD symptom (= PCL-5 item) columns*

Description

Standardizes column names in PCL-5 (PTSD Checklist for DSM-5) data by renaming them to a consistent format (symptom_1 through symptom_20). This standardization is essential for subsequent analyses using other functions in the package.

Usage

```
rename_ptsd_columns(data, id_col = NULL)
```

Arguments

`data`	A dataframe containing exactly 20 columns of PCL-5 item scores (plus any columns named in `id_col` for carry-through). The scores should be on a 0-4 scale where: • 0 = Not at all • 1 = A little bit • 2 = Moderately • 3 = Quite a bit • 4 = Extremely

id_col — Optional character vector naming column(s) in data to preserve as identifiers. These columns are kept alongside the renamed symptom columns and propagate through the rest of the workflow (optimize_combinations, apply_symptom_combinations, holdout_validation, cross_validation). Use them as a join key to merge per-row diagnoses back to the original dataframe (e.g. demographics). Defaults to NULL (no carry-through; all non-symptom columns are dropped).

Details

The function assumes the input data contains exactly 20 columns corresponding to the 20 items of the PCL-5. The columns are renamed sequentially from symptom_1 to symptom_20, maintaining their original order. The PCL-5 items correspond to different symptom clusters:

- symptom_1 to symptom_5: Intrusion symptoms (Criterion B)
- symptom_6 to symptom_7: Avoidance symptoms (Criterion C)
- symptom_8 to symptom_14: Negative alterations in cognitions and mood (Criterion D)
- symptom_15 to symptom_20: Alterations in arousal and reactivity (Criterion E)

Value

A dataframe with PCL-5 columns renamed to symptom_1 through symptom_20. If id_col is supplied, the named columns are prepended (in original order).

See Also

check_pcl5_data for a pre-flight check that reports every data problem at once before this step.

Examples

```
# Example with a sample PCL-5 dataset
sample_data <- data.frame(
  matrix(sample(0:4, 20 * 10, replace = TRUE),
         nrow = 10,
         ncol = 20)
)
renamed_data <- rename_ptsd_columns(sample_data)
colnames(renamed_data)  # Shows new column names

# Carry a participant identifier through the workflow
sample_data$patient_id <- sprintf("P%03d", seq_len(nrow(sample_data)))
with_id <- rename_ptsd_columns(sample_data, id_col = "patient_id")
head(with_id)
```

score_all_combinations *Score every candidate symptom combination*

Description

Scores **every** candidate combination of `n_symptoms` PCL-5 items against the full DSM-5-TR diagnosis and returns the complete ranked table – not just the best ones. This is the exhaustive companion to `optimize_combinations` / `optimize_combinations_clusters` (which keep only the top `n_top`): use it to study how performance decays across the whole candidate set, e.g. to show that many symptom sets are near-interchangeable (a plateau of near-optimal combinations followed by a drop).

Usage

```
score_all_combinations(
  data,
  n_symptoms = 6,
  n_required = 4,
  clusters = NULL,
  score_by = "balanced_accuracy",
  chunk_size = 1000,
  show_progress = TRUE
)
```

Arguments

`data`	A dataframe with the 20 PCL-5 item columns `symptom_1` through `symptom_20` (output of `rename_ptsd_columns`). Additional carry-through columns are ignored.
`n_symptoms`	Integer. Number of items per combination (default 6).
`n_required`	Integer. How many of the items must be present (score >= 2) for a positive diagnosis (default 4).
`clusters`	`NULL` (default) to score all subsets without a cluster constraint, or a named list of integer vectors (e.g. `list(B = 1:5, C = 6:7, D = 8:14, E = 15:20)`) for the cluster-constrained candidate set and diagnosis rule.
`score_by`	Character. Metric that defines the ranking: `"balanced_accuracy"` (default), `"accuracy"`, or `"sensitivity"`. All metrics are returned regardless; this only sets the sort order.
`chunk_size`	Integer. Number of combinations scored per chunk (default 1000). Affects speed and parallel granularity only, never the result.
`show_progress`	Logical. If `TRUE` (default), display a progress bar (sequential mode only).

Details

With clusters = NULL, all choose(20, n_symptoms) subsets are scored (38,760 for six symptoms). With a cluster structure, only the combinations containing at least one item per cluster are scored (13,685 six-symptom sets for the default PCL-5 clusters), and the diagnosis additionally requires the present symptoms to span all clusters – the same candidate set and rule as optimize_combinations_clusters. The hierarchical per-row cluster check makes this mode noticeably slower.

Combinations are processed in chunks. If the **future.apply** package is installed and a future plan is set (e.g. future::plan(future::multisession)), chunks are scored in parallel; otherwise they are scored sequentially with a progress bar. Results are identical either way.

The returned combination_id uses the same canonical format as write_combinations (sorted item numbers joined by underscores), so the full curve can be joined against exported top-k combinations.

Value

A data.frame with one row per candidate combination, sorted best-to-worst by score_by (ties broken by combination_id for determinism):

- rank: 1 = best.
- combination_id: sorted item numbers joined by underscores (e.g. "1_6_8_10_15_19").
- tp, fn, fp, tn: the 2x2 counts against the full DSM-5-TR diagnosis.
- sensitivity, specificity, ppv, npv, accuracy, balanced_accuracy: metrics on the 0-1 scale (NA where a denominator is zero).

The attributes n_symptoms, n_required, clusters, score_by, and n_combinations record the configuration.

See Also

optimize_combinations, optimize_combinations_clusters, compare_optimizations.

Examples

```
# A 4-symptom search on a 250-row subset keeps the example fast
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))
curve <- score_all_combinations(ptsd_data, n_symptoms = 4, n_required = 3,
                                show_progress = FALSE)
nrow(curve)     # choose(20, 4) = 4845 combinations, all ranked
head(curve)

# The full balanced-accuracy curve, e.g. for a rank plot:
# plot(curve$rank, curve$balanced_accuracy, type = "l", log = "x")
```

set_id_to_items *Convert between combination ids and item numbers*

Description

A combination id is the item numbers joined by underscores, which is the form `score_all_combinations` and `compute_plateau` return.

Usage

```
set_id_to_items(set_id)

format_set_items(set_id)

format_set_labels(set_id)
```

Arguments

`set_id`	Character vector of combination ids, such as `"2_3_6_7_17_18"`.

Value

`set_id_to_items()` gives a list of integer vectors; `format_set_items()` a character vector such as `"2, 3, 6, 7, 17, 18"`; `format_set_labels()` the item labels, semicolon separated.

Examples

```
set_id_to_items("2_3_6_7_17_18")
format_set_items("2_3_6_7_17_18")
format_set_labels("2_3_6_7_17_18")
```

simulated_ptsd *Simulated PCL-5 (PTSD Checklist) Data*

Description

A dataset containing simulated responses from 5,000 patients on the PCL-5 (PTSD Checklist for DSM-5). Each patient rated 20 PTSD symptoms on a scale from 0 to 4. The dataset additionally ships with three demographic columns (`patient_id`, `age`, `sex`) so that the demographic carry-through workflow (`id_col`) can be demonstrated end-to-end on the bundled data.

Usage

```
simulated_ptsd
```

Format

A data frame with 5,000 rows and 23 columns:

patient_id Character. Synthetic participant identifier (`"P0001"`–`"P5000"`).

age Integer. Age in years (truncated normal, range 18–80).

sex Factor. `"female"` / `"male"`.

S1 Intrusive memories

S2 Nightmares

S3 Flashbacks

S4 Emotional reactivity to reminders

S5 Physical reactions to reminders

S6 Avoiding memories/thoughts/feelings

S7 Avoiding external reminders

S8 Amnesia

S9 Strong negative beliefs

S10 Distorted blame

S11 Negative trauma-related emotions

S12 Decreased interest in activities

S13 Detachment or estrangement

S14 Trouble experiencing positive emotions

S15 Irritability/aggression

S16 Risk-taking behavior

S17 Hypervigilance

S18 Heightened startle reaction

S19 Difficulty concentrating

S20 Sleep problems

Details

The symptoms are rated on a 5-point scale:

- 0 = Not at all
- 1 = A little bit
- 2 = Moderately
- 3 = Quite a bit
- 4 = Extremely

The symptoms correspond to DSM-5 PTSD criteria:

- Symptoms 1-5: Criterion B (Intrusion)
- Symptoms 6-7: Criterion C (Avoidance)
- Symptoms 8-14: Criterion D (Negative alterations in cognitions and mood)
- Symptoms 15-20: Criterion E (Alterations in arousal and reactivity)

Note

Symptoms were simulated independently. Real PCL-5 data exhibits within-cluster correlations (e.g., between intrusion symptoms). Optimization results on these data are for illustration only; real-world performance should be evaluated on empirical datasets.

Source

Simulated data for demonstration purposes

---

simulated_ptsd_genpop *Simulated General Population PCL-5 Data*

Description

A dataset containing simulated responses from 1,200 individuals in a general population sample on the PCL-5 (PTSD Checklist for DSM-5). Each individual rated 20 PTSD symptoms on a scale from 0 to 4. This dataset has a lower PTSD prevalence (~21 simulated_ptsd (~94 validation of diagnostic criteria derived from clinical samples. Like simulated_ptsd, it ships with three demographic columns (patient_id, age, sex). It also ships paired clinician-administered CAPS-5 severity ratings (C1–C20) for the same participants, simulated to correlate with the PCL-5 items so that the PCL-5 and CAPS-5 total scores correlate about 0.8 (the level usually reported empirically). This supports the paired-instrument workflow in the CAPS-5 vignette without inventing data inline.

Usage

```
simulated_ptsd_genpop
```

Format

A data frame with 1,200 rows and 43 columns:

patient_id Character. Synthetic participant identifier ("G0001"–"G1200").

age Integer. Age in years (truncated normal, range 18–80).

sex Factor. "female" / "male".

S1 PCL-5: Intrusive memories

S2 PCL-5: Nightmares

S3 PCL-5: Flashbacks

S4 PCL-5: Emotional reactivity to reminders

S5 PCL-5: Physical reactions to reminders

S6 PCL-5: Avoiding memories/thoughts/feelings

S7 PCL-5: Avoiding external reminders

S8 PCL-5: Amnesia

S9 PCL-5: Strong negative beliefs

S10 PCL-5: Distorted blame

S11 PCL-5: Negative trauma-related emotions

S12 PCL-5: Decreased interest in activities

S13 PCL-5: Detachment or estrangement

S14 PCL-5: Trouble experiencing positive emotions

S15 PCL-5: Irritability/aggression

S16 PCL-5: Risk-taking behavior

S17 PCL-5: Hypervigilance

S18 PCL-5: Heightened startle reaction

S19 PCL-5: Difficulty concentrating

S20 PCL-5: Sleep problems

C1 CAPS-5: Intrusive memories

C2 CAPS-5: Nightmares

C3 CAPS-5: Flashbacks

C4 CAPS-5: Emotional reactivity to reminders

C5 CAPS-5: Physical reactions to reminders

C6 CAPS-5: Avoiding memories/thoughts/feelings

C7 CAPS-5: Avoiding external reminders

C8 CAPS-5: Amnesia

C9 CAPS-5: Strong negative beliefs

C10 CAPS-5: Distorted blame

C11 CAPS-5: Negative trauma-related emotions

C12 CAPS-5: Decreased interest in activities

C13 CAPS-5: Detachment or estrangement

C14 CAPS-5: Trouble experiencing positive emotions

C15 CAPS-5: Irritability/aggression

C16 CAPS-5: Risk-taking behavior

C17 CAPS-5: Hypervigilance

C18 CAPS-5: Heightened startle reaction

C19 CAPS-5: Difficulty concentrating

C20 CAPS-5: Sleep problems

The CAPS-5 items (`C1`–`C20`) are paired clinician-administered severity ratings for the same participants and the same symptoms as the PCL-5 items, simulated to correlate with them at a total-score r of about 0.8.

Details

The symptoms are rated on a 5-point scale:

- 0 = Not at all
- 1 = A little bit
- 2 = Moderately
- 3 = Quite a bit
- 4 = Extremely

The symptoms correspond to DSM-5 PTSD criteria:

- Symptoms 1-5: Criterion B (Intrusion)
- Symptoms 6-7: Criterion C (Avoidance)
- Symptoms 8-14: Criterion D (Negative alterations in cognitions and mood)
- Symptoms 15-20: Criterion E (Alterations in arousal and reactivity)

The data was simulated as a mixture of approximately 17% individuals with elevated symptom profiles (PTSD-like) and 83% with low symptom levels, resulting in approximately 21% meeting full DSM-5 diagnostic criteria.

Note

Symptoms were simulated independently. Real PCL-5 data exhibits within-cluster correlations (e.g., between intrusion symptoms). Optimization results on these data are for illustration only; real-world performance should be evaluated on empirical datasets.

Source

Simulated data for demonstration purposes

See Also

`simulated_ptsd` for the clinical veteran sample with higher prevalence.

`summarize_ptsd` *Summarize PTSD scores and diagnoses*

Description

Creates a summary of PCL-5 total scores and PTSD diagnoses, including mean total score, standard deviation, and number of positive diagnoses.

Usage

```
summarize_ptsd(data)
```

Arguments

`data` A dataframe containing at minimum:

- A 'total' column with PCL-5 total scores (from calculate_ptsd_total)
- A 'PTSD_orig' column with TRUE/FALSE values (from determine_ptsd_diagnosis)

Details

This function calculates key summary statistics for PCL-5 data:

- Mean total score (severity indicator)
- Standard deviation of total scores (variability in severity)
- Count of positive PTSD diagnoses (prevalence in the sample)

Value

A dataframe with one row containing:

- mean_total: Mean PCL-5 total score
- sd_total: Standard deviation of PCL-5 total scores
- n_diagnosed: Number of positive PTSD diagnoses

Examples

```
# Create sample data
sample_data <- data.frame(
  total = sample(0:80, 100, replace = TRUE),
  PTSD_orig = sample(c(TRUE, FALSE), 100, replace = TRUE)
)

# Generate summary statistics
summary_stats <- summarize_ptsd(sample_data)
print(summary_stats)
```

`summarize_ptsd_changes` *Summarize changes in PTSD diagnostic metrics*

Description

Compares different PTSD diagnostic criteria by calculating diagnostic accuracy metrics and changes in diagnosis status relative to a baseline criterion.

Usage

```
summarize_ptsd_changes(data)
```

Arguments

`data` A dataframe where:

- Each column represents a different diagnostic criterion
- Must include a column named "PTSD_orig" as the baseline criterion
- Values are logical (TRUE/FALSE) indicating whether PTSD criteria are met
- Each row represents one case/participant

Details

The function calculates multiple diagnostic metrics comparing each diagnostic criterion to a baseline criterion (PTSD_orig):

Basic counts:

- Number and percentage of diagnosed/non-diagnosed cases per criterion
- Number of newly diagnosed (false positive) and newly non-diagnosed (false negative) cases
- True positive and true negative cases

Diagnostic accuracy metrics:

- Sensitivity: Proportion of true PTSD cases correctly identified
- Specificity: Proportion of non-PTSD cases correctly identified
- PPV (Positive Predictive Value): Probability that a positive diagnosis is correct
- NPV (Negative Predictive Value): Probability that a negative diagnosis is correct

Value

A dataframe containing the following columns for each diagnostic criterion:

- column: Name of the diagnostic criterion
- diagnosed: Number of cases diagnosed as PTSD
- non_diagnosed: Number of cases not diagnosed as PTSD
- diagnosed_percent: Percentage of cases diagnosed
- non_diagnosed_percent: Percentage of cases not diagnosed
- newly_diagnosed: Cases diagnosed under new but not baseline criterion (false positive)
- newly_nondiagnosed: Cases diagnosed under baseline but not new criterion (false negative)
- true_positive: Cases diagnosed under both criteria
- true_negative: Cases not diagnosed under either criterion
- true_cases: Sum of true positives and true negatives
- false_cases: Sum of newly diagnosed (false positive) and newly non-diagnosed (false negative)
- sensitivity, specificity, ppv, npv: Standard diagnostic accuracy metrics
- accuracy: Proportion of cases classified the same as the baseline criterion ((true positives + true negatives) / total)
- balanced_accuracy: Mean of sensitivity and specificity ((sensitivity + specificity) / 2), robust to imbalanced prevalence of the baseline diagnosis

Examples

```
# Create sample diagnostic data
set.seed(123)
n_cases <- 100
sample_data <- data.frame(
  PTSD_orig = sample(c(TRUE, FALSE), n_cases, replace = TRUE),
  PTSD_alt1 = sample(c(TRUE, FALSE), n_cases, replace = TRUE),
  PTSD_alt2 = sample(c(TRUE, FALSE), n_cases, replace = TRUE)
)

# Calculate diagnostic metrics
diagnostic_metrics <- summarize_ptsd_changes(sample_data)
diagnostic_metrics
```

---

summarize_top_combinations

Build a tidy comparison table of top combinations across scenarios

---

Description

Produces a manuscript-ready table summarising the diagnostic performance of each top combination (or fixed criterion) in a `compare_optimizations` result. The output matches the layout of the PTSDdiag preprint's Table 2: one row per combination, with Approach / Rank / Combination / TP / FN / FP / TN / Sensitivity / Specificity / PPV / NPV / Accuracy / Balanced Accuracy.

Usage

```
summarize_top_combinations(comparison, top_n = NULL, as_percent = FALSE)
```

Arguments

`comparison`	A `ptsdiag_comparison` object.
`top_n`	Optional integer. Per-scenario limit on combinations to include. Fixed scenarios always contribute exactly one row. Default `NULL` returns all stored combinations.
`as_percent`	Logical. If `TRUE`, Sensitivity/Specificity/PPV/NPV/Accuracy/Balanced Accuracy are returned as percentages (0-100); otherwise as fractions (0-1). Default `FALSE`.

Details

For each scenario, the per-row `diagnosis_comparison` dataframe is summarised via `summarize_ptsd_changes`. The self-comparison `PTSD_orig` row is dropped, the remaining rows are renamed, and the scenario label is prepended.

Sensitivity, specificity, PPV, NPV, accuracy and balanced accuracy are returned on the 0-1 fraction scale by default (matching `compare_diagnostic_systems`); set `as_percent = TRUE` to convert to 0-100 for manuscript display. Accuracy is `(TP + TN) / N`, the quantity maximised by `score_by = "accuracy"`; balanced accuracy is `(sensitivity + specificity) / 2`, the quantity maximised by the default `score_by = "balanced_accuracy"`.

Value

A data.frame with columns: Approach, Rank, Combination, TP, FN, FP, TN, Sensitivity, Specificity, PPV, NPV, Accuracy, Balanced Accuracy.

See Also

compare_optimizations, symptom_frequency, plot_symptom_frequency.

Examples

```
# Use a 250-row subset and a small 4-symptom search to keep the example
# fast; omit `scenarios` to run the three default rules
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))
comp <- compare_optimizations(
  ptsd_data,
  scenarios = list(
    "3/4 Non-hierarchical" = list(n_symptoms = 4, n_required = 3,
                                  hierarchical = FALSE)
  ),
  include_icd11 = TRUE, n_top = 5, show_progress = FALSE
)
summarize_top_combinations(comp, as_percent = TRUE)
```

---

symptom_frequency *Per-symptom inclusion counts across optimization scenarios*

Description

Returns a long-format data frame giving how often each of the 20 PCL-5 symptoms appears in the top combinations of each scenario in a compare_optimizations result. This is the data source for plot_symptom_frequency and matches the structure of the preprint's Supplementary Table S4.

Usage

```
symptom_frequency(
  comparison,
  include_overall = TRUE,
  overall_includes_fixed = FALSE,
  clusters = NULL
)
```

Arguments

comparison	A ptsdiag_comparison object.
include_overall	Logical. If TRUE (default), an OVERALL row is appended that pools across scenarios.

`overall_includes_fixed`
: Logical. If `TRUE`, fixed criteria contribute to the OVERALL row. Default `FALSE`.

`clusters`
: Named list of item indices defining the clusters used by the hierarchical scenarios, for the `Baseline` column. Defaults to the standard DSM-5-TR PCL-5 clusters.

Details

For optimize scenarios, `Count` ranges from 0 to `n_top` (the number of stored combinations). For fixed scenarios such as ICD-11, the fixed symptom set contributes exactly one combination so `Count` is either 0 or 1. `RelFreq` normalises `Count` by the number of combinations stored in that scenario.

The optional `OVERALL` row pools counts across scenarios. By default fixed scenarios are excluded from the OVERALL pool so that OVERALL continues to reflect data-driven symptom selection. Set `overall_includes_fixed = TRUE` to weight every combination equally.

`RelFreq` cannot be read on its own. Every combination contains the same number of items, so the frequencies are zero-sum and an item can only gain at another's expense. `Baseline` gives the proportion of the scenario's own candidate space that contains each item, and `Enrichment` their ratio: values above 1 mean the item was selected more often than drawing combinations at random would produce. The baseline matters most under the hierarchical rule, where it varies by cluster because every combination must draw at least one item from each. Fixed criteria such as ICD-11 have no candidate space, so both columns are `NA` for them.

Value

A data.frame with columns `Symptom` (integer 1-20), `Approach` (factor with levels in scenario order, optionally ending in `"OVERALL"`), `Count` (integer), `RelFreq` (numeric in \[0, 1\]), `Baseline` (the chance selection frequency for that scenario, `NA` for fixed criteria) and `Enrichment` (`RelFreq / Baseline`).

See Also

`compare_optimizations`, `plot_symptom_frequency`, `summarize_top_combinations`.

Examples

```
# Use a 250-row subset and a small 4-symptom search to keep the example
# fast; omit `scenarios` to run the three default rules
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:250, ],
                                 id_col = c("patient_id", "age", "sex"))
comp <- compare_optimizations(
  ptsd_data,
  scenarios = list(
    "3/4 Non-hierarchical" = list(n_symptoms = 4, n_required = 3,
                                  hierarchical = FALSE)
  ),
  include_icd11 = TRUE, n_top = 5, show_progress = FALSE
)
freq <- symptom_frequency(comp)
head(freq)
```

`symptom_selection` *Symptom selection across the plateau, against chance*

Description

Reports how often each PCL-5 item appears among the near-optimal symptom combinations, alongside the frequency expected by chance alone. Raw selection counts cannot be read on their own: every combination contains the same number of items, so the counts are zero-sum and an item can only gain at another's expense. The chance baseline turns "item 6 appears in 55 plateau combinations" into a statement about whether item 6 is actually being preferred.

Usage

```
symptom_selection(
  fit,
  delta = 1,
  stability = NULL,
  clusters = NULL,
  symptom_labels = NULL
)
```

Arguments

`fit`	A fitted exhaustive search from `score_all_combinations`.
`delta`	Numeric. Plateau width in **percentage points** of balanced accuracy (default 1). A single value.
`stability`	Optional `bootstrap_stability` result for the same search, used for the stability weighted columns.
`clusters`	Named list of item indices used only to label items by cluster in the output. Defaults to the clusters of `fit`, or to the standard DSM-5-TR PCL-5 clusters when `fit` was non-hierarchical.
`symptom_labels`	Optional character vector of length 20 giving readable item names.

Details

The baseline is the proportion of the rule form's own candidate space that contains each item, computed by enumeration rather than assumed. Under the non-hierarchical rule it is the same for every item (`n_symptoms / 20`, so 30 hierarchical rule it varies substantially by cluster, because a combination must draw at least one item from each: with the default PCL-5 clusters, each of the two avoidance items appears in 55.8 each of the seven cognition-and-mood items appears in only 25.2 hierarchical selection frequencies against a flat 30 avoidance items look preferred when they are merely unavoidable.

Supplying a `bootstrap_stability` result adds a stability weighted version of the same quantity, in which each combination counts in proportion to how often it stayed on the plateau across bootstrap replicates rather than counting once if it happened to make the plateau in the original sample.

Value

A data.frame with one row per item:

- `symptom`, `label`, `cluster`: item identity.
- `baseline`: proportion of this rule form's candidate space containing the item.
- `plateau_count`, `plateau_n`, `plateau_freq`: how often the item appears among the plateau combinations.
- `enrichment`: `plateau_freq / baseline`. Values above 1 mean the item is selected more often than chance would produce.
- `excess_pp`: `plateau_freq - baseline` in percentage points.
- `pi_freq`, `pi_enrichment`: the stability weighted equivalents, or `NA` when `stability` is not supplied.

See Also

`compute_plateau`, `bootstrap_stability`, `plot_symptom_selection`.

Examples

```
ptsd_data <- rename_ptsd_columns(simulated_ptsd[1:120, ],
                                 id_col = c("patient_id", "age", "sex"))
fit <- score_all_combinations(ptsd_data, n_symptoms = 4, n_required = 3,
                              show_progress = FALSE)

selection <- symptom_selection(fit, delta = 1)
head(selection[order(-selection$enrichment), ])
```

---

transport_plateau *Summarise a transported plateau for release*

Description

The releasable counterpart of `evaluate_sets`. A site that holds data it cannot share receives a list of symptom combinations derived elsewhere, evaluates them locally, and returns two things and nothing else:

- a five-number summary of balanced accuracy, sensitivity and specificity **across** each transported plateau – order statistics over combinations, with the combination-to-value linkage discarded;
- two-by-two counts for a short, pre-specified list of named combinations.

Usage

```
transport_plateau(
  data,
  plateau_specs,
  named_specs,
  sample_id,
  payload_id = NA_character_,
  flag_cell = 5L
)
```

Arguments

data	A data frame with symptom_1 through symptom_20.
plateau_specs	A named list of plateau specifications. Each element is a list with derivation, rule_form, delta, n_required, clusters and sets.
named_specs	A named list of the same shape, each additionally carrying ranks alongside sets.
sample_id	Identifier of this sample. It names every returned file and appears as a column in every returned table.
payload_id	Identifier of the payload these combinations came from, carried through to the returns so a bundle can be matched to the request that produced it. Optional.
flag_cell	Warn about any two-by-two cell below this size.

Details

Why the linkage is discarded. Per-combination (tp, fp, fn, tn) for a whole candidate space of 38,760 six-item subsets is 77,520 linear constraints on the response-pattern count vector, which in a sample of a few hundred has at most a few hundred non-zero entries among 2^20. That is an over-determined sparse-recovery problem, so the multiset of binarized response patterns cross-classified by reference label would be recoverable – and at that size a rare pattern is effectively an individual record. A five-number summary over combinations supports no such reconstruction.

This function therefore has no argument that emits per-combination detail for a plateau. That is deliberate: a collaborator cannot release it by accident.

Counts for the named combinations are **not** suppressed when small, because a confusion-matrix table of a handful of pre-specified rules is standard reporting and blanking cells would empty it. Instead flag_cell warns, so a small-cell disclosure is visible before the bundle is sent.

Value

An object of class ptsdiag_transport: a list of summary, named, sample_id and payload_id. Because the point of this object is that its contents are exactly what may be released, both tables are listed in full.

- summary: one row per plateau specification and metric, with sample_id, payload_id, derivation, rule_form, delta, metric ("ba", "sensitivity", "specificity"), and the five-number summary n_sets, min, q1, median, q3, max. No combination identifiers appear here.

- named: one row per named combination plus one for ICD-11, with sample_id, payload_id, derivation, rule_form, rank, set_id, the counts tp, fn, fp, tn, and every non-count column of diagnostic_metrics computed from them (n, n_pos, n_neg, prevalence, sensitivity, specificity, ppv, npv, accuracy, ba, lr_pos, lr_neg, kappa, each with _lo and _hi Wilson bounds where applicable, and pct_agreement).

See Also

write_transport to serialise, evaluate_sets for the per-combination results that stay local.

Examples

```
ptsd_data <- rename_ptsd_columns(
  simulated_ptsd_genpop[1:400, c("patient_id", paste0("S", 1:20))],
  id_col = "patient_id")
fit <- score_all_combinations(ptsd_data, n_symptoms = 3, n_required = 2,
                              show_progress = FALSE)
plateau <- compute_plateau(fit, delta = 1)

bundle <- transport_plateau(
  ptsd_data,
  plateau_specs = list(list(derivation = "example", rule_form = "flat_2_3",
                            delta = 1, n_required = 2, clusters = NULL,
                            sets = plateau$sets$combination_id)),
  named_specs = list(list(derivation = "example", rule_form = "flat_2_3",
                          n_required = 2, clusters = NULL,
                          sets = fit$combination_id[1:3], ranks = 1:3)),
  sample_id = "site_a"
)
bundle
bundle$summary
```

---

wilson_ci *Wilson score interval for a binomial proportion*

Description

The default interval for a proportion here. It behaves sensibly at zero and at one, where the Wald interval does not.

Usage

```
wilson_ci(x, n, conf_level = 0.95)
```

Arguments

x	Number of successes. Vectorised.
n	Number of trials. Vectorised.
conf_level	Confidence level.

Value

A two-column matrix with columns `lo` and `hi`, on the 0-1 scale.

Examples

```
wilson_ci(80, 100)
```

`write_combinations` *Write symptom combinations to a JSON file*

Description

Exports optimized symptom combinations to a human-readable JSON file. This enables sharing of derived combinations across research groups without needing to share raw data, supporting reproducible derivation-validation workflows.

Usage

```
write_combinations(
  combinations,
  file,
  n_required = 4,
  clusters = NULL,
  n_symptoms = NULL,
  score_by = NULL,
  description = "",
  label = NULL
)
```

Arguments

`combinations`	A list of integer vectors specifying symptom combinations, or the full result object from `optimize_combinations` / `optimize_combinations_clusters` (which contains `$best_symptoms`).
`file`	Character string. Path to the output JSON file.
`n_required`	Integer specifying how many symptoms must be present for a positive diagnosis (default: 4). This value is stored in the file so that `read_combinations` can retrieve it.
`clusters`	`NULL` (default) for non-hierarchical combinations, or a named list of integer vectors defining the cluster structure for hierarchical combinations. Stored in the file for use with `apply_symptom_combinations`.
`n_symptoms`	Integer or `NULL`. Number of symptoms per combination. If `NULL` (default), inferred from the length of the first combination.
`score_by`	Character or `NULL`. The scoring criterion used during optimization (`"balanced_accuracy"`, `"accuracy"`, or `"sensitivity"`). Stored as metadata for reproducibility. If `NULL` (default), omitted from the file.

description	Character string. Optional free-text description of the derivation context (e.g., sample characteristics, dataset name). Default is an empty string.
label	Character string or NULL (default). Optional short display label for the rule these combinations implement (e.g. "4/6 Hierarchical"). Stored in the file and used by as_definitions to name the definition at the validation site, so the derivation site controls how the rule is labelled in downstream tables.

Details

The JSON file contains the combinations alongside metadata needed to apply them via apply_symptom_combinations: the required symptom threshold (n_required) and optional cluster structure (clusters). Additional fields (score_by, n_symptoms, description) provide context for reproducibility.

The combinations argument can be either:

- A list of integer vectors (e.g., from results$best_symptoms)
- The full result object from optimize_combinations or optimize_combinations_clusters (the function automatically extracts $best_symptoms)

Value

The file path (invisibly), following the convention of write.csv.

See Also

read_combinations to import combinations from a JSON file.

optimize_combinations and optimize_combinations_clusters for deriving optimal combinations.

apply_symptom_combinations for applying imported combinations to new data.

Examples

```
# Create example combinations
my_combos <- list(
  c(1, 6, 8, 10, 15, 19),
  c(2, 7, 9, 11, 16, 20)
)

# Write to a temporary file
tmp <- tempfile(fileext = ".json")
write_combinations(my_combos, tmp, n_required = 4,
                   score_by = "balanced_accuracy",
                   description = "Example non-hierarchical combinations")

# Can also pass a full optimization result directly:
# write_combinations(optimization_result, tmp, n_required = 4)

# Clean up
unlink(tmp)
```

`write_transport` *Write a return bundle*

Description

Serialises exactly the files a site is expected to return, and nothing else. Every filename is prefixed with the sample identifier, so bundles from different sites remain distinguishable when they are loose files in one directory rather than in separate folders.

Usage

```
write_transport(
  x,
  dir,
  sample_info = NULL,
  flow = NULL,
  caps_agreement = NULL,
  session_info = FALSE
)
```

Arguments

`x`	A `ptsdiag_transport` object.
`dir`	Directory to write into. Required, and must already exist: the function never creates directories and never guesses a location, so a mistyped path fails rather than scattering files.
`sample_info`, `flow`	Named lists of aggregate descriptives, or `NULL`.
`caps_agreement`	A data frame of counts against a clinician interview, or `NULL`.
`session_info`	Whether to write `sessionInfo()` alongside. `FALSE` by default, so nothing about the local machine leaves it unless you ask.

Details

`sample_info` and `flow` must be plain lists of aggregates. They are written as JSON deliberately: a **gtsummary** or **ggplot2** object would carry a reference to the source data inside it.

Value

The paths written, invisibly.

See Also

`read_transport`, `transport_plateau`.

Examples

```
ptsd_data <- rename_ptsd_columns(
  simulated_ptsd_genpop[1:400, c("patient_id", paste0("S", 1:20))],
  id_col = "patient_id")
fit <- score_all_combinations(ptsd_data, n_symptoms = 3, n_required = 2,
                              show_progress = FALSE)
plateau <- compute_plateau(fit, delta = 1)
bundle <- transport_plateau(
  ptsd_data,
  plateau_specs = list(list(derivation = "example", rule_form = "flat_2_3",
                            delta = 1, n_required = 2, clusters = NULL,
                            sets = plateau$sets$combination_id)),
  named_specs = list(list(derivation = "example", rule_form = "flat_2_3",
                          n_required = 2, clusters = NULL,
                          sets = fit$combination_id[1:3], ranks = 1:3)),
  sample_id = "site_a"
)

# Write into a temporary directory, then clean up
out <- file.path(tempdir(), "transport_example")
dir.create(out, showWarnings = FALSE)
write_transport(bundle, out)
list.files(out)
unlink(out, recursive = TRUE)
```

Index