

Package ‘semanticfa’

September 1, 2026

Title Semantic Factor Analysis of Language Model Embeddings

Version 0.5.0

Description Performs exploratory factor analysis on language model embeddings of psychological scale items. Embeds item text with sentence transformers or other language models, transforms the embeddings into item-by-item similarity matrices, and extracts latent factor structure via standard exploratory factor analysis, using several similarity transforms (atomic reversed, SQuID centering, mean-centered Pearson) and fit diagnostics tailored to embedding matrices (TEFI, RMSR, CAF, McDonald's omega). Factor retention spans embedding-adapted parallel analysis, the empirical Kaiser criterion, Velicer's minimum average partial, a comparison-data misfit profile, and a calibrated learned rule that reports conformal intervals. Further tools orient factor axes toward retrieved construct terms by lexical target rotation, and audit whether a scale's items cover their construct's semantic region without collecting responses. The underlying methods are documented with full citations in the corresponding function help pages. Returns objects compatible with 'psych' and 'EFAtools' workflows.

License GPL (>= 3)

URL <https://github.com/devon7y/semanticfa>

BugReports <https://github.com/devon7y/semanticfa/issues>

Depends R (>= 4.1.0)

Imports digest, GPArotation, grDevices, graphics, psych, reticulate (>= 1.41.0), Rtsne, stats, utils, uwot, withr

Suggests EFAtools, jsonlite, EGAnet, httr2, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Devon Yanitski [aut, cre] (ORCID:
<https://orcid.org/0009-0006-1568-3387>),
 Chris Westbury [aut]

Maintainer Devon Yanitski <dyanitsk@ualberta.ca>

Repository CRAN

Date/Publication 2026-09-01 21:00:02 UTC

Contents

semanticfa-package	3
as_psych	4
big5	5
plot.sfa_coverage	6
plot.sfa_coverage_battery	6
sfa	7
sfa_anchor	10
sfa_build_bank	11
sfa_build_region	12
sfa_build_regions	14
sfa_cd	16
sfa_clear_cache	18
sfa_combine_banks	18
sfa_congruence	19
sfa_corplot	20
sfa_coverage	21
sfa_cross_matrix	25
sfa_deletion_gaps	26
sfa_dimselect	27
sfa_ekc	29
sfa_embed	30
sfa_embedding_bank	31
sfa_gaps	32
sfa_install_python	32
sfa_itemplot	33
sfa_item_fit	35
sfa_jinglejangle	36
sfa_leximax	38
sfa_lexmap	40
sfa_load_npz	41
sfa_load_region	42
sfa_map	42
sfa_name	43
sfa_nameability	45
sfa_naming_instruction	46
sfa_nfactors	46

sfa_nli_matrix	47
sfa_parallel	49
sfa_pool	50
sfa_project	51
sfa_redundancy	53
sfa_reembed_region	54
sfa_region_bank	55
sfa_semk	56
sfa_similarity	58
sfa_simplify	59

Index	62
--------------	-----------

semanticfa-package	<i>semanticfa: Semantic Factor Analysis of Language Model Embeddings</i>
--------------------	--

Description

Recovers the latent factor structure of a psychological scale from the *meaning of its item wording* — no human response data required. It embeds item text with a language model, turns the embeddings into an item-by-item similarity matrix, and runs exploratory factor analysis, with a suite of tools for inspecting and refining the scale.

Main entry point

- [sfa](#) — run the full pipeline (embed -> similarity -> retention -> extraction -> diagnostics) and return an "sfa" object with print, summary, plot, and [as_psych](#) methods.

Building blocks

- [sfa_embed](#), [sfa_install_python](#) — turn item text into embeddings.
- [sfa_similarity](#) — similarity transforms / encodings (atomic, atomic-reversed, SQuID, mean-centered Pearson).
- [sfa_parallel](#), [sfa_nfactors](#), [sfa_dimselect](#) — choose the number of factors and which embedding dimensions to use.

Item- and scale-level tools

- [sfa_anchor](#) — item-by-construct belonging (a semantic loading table).
- [sfa_redundancy](#) — detect near-duplicate items.
- [sfa_simplify](#) — build response-free short forms.
- [sfa_project](#) — place items on a named bipolar axis (e.g. mild -> severe).
- [sfa_jinglejangle](#) — compare whole scales for jingle/jangle fallacies.
- [sfa_nli_matrix](#) — valence-aware (entailment vs. contradiction) similarity.
- [sfa_congruence](#) — compare the recovered structure to theory or empirical data.

Example data

[big5](#) — IPIP Big-Five 50-item markers with precomputed embeddings, used throughout the examples.

Author(s)

Authors:

- **Devon Yanitski** (author, maintainer) <dyanitsk@ualberta.ca> ([ORCID](#))
- Chris Westbury (author)

See Also

Useful links:

- <https://github.com/devon7y/semanticfa>
- Report bugs at <https://github.com/devon7y/semanticfa/issues>

as_psych

Coerce to psych fa Object

Description

Coerce to psych fa Object

Usage

```
as_psych(x, ...)

## S3 method for class 'sfa'
as_psych(x, ...)
```

Arguments

x	An object to coerce.
...	Additional arguments (unused).

Value

An object of class `c("psych", "fa")`.

big5	<i>IPIP Big Five 50-Item Inventory with Qwen3 Embeddings</i>
------	--

Description

A bundled example dataset containing the 50-item IPIP Big Five personality inventory with precomputed sentence-embedding vectors. The scale has 5 factors (Extraversion, Agreeableness, Conscientiousness, Neuroticism, Openness) with 10 items each, including 18 reverse-keyed items, making it suitable for demonstrating all encoding methods.

Usage

big5

Format

A list with components:

- items** Character vector (length 50): item text.
- codes** Character vector (length 50): item codes (E1, E2, ..., O50).
- factors** Character vector (length 50): theoretical factor labels.
- scoring** Numeric vector (length 50): +1 or -1 keying direction.
- embeddings** Numeric matrix (50 x 4096): precomputed embeddings from the Qwen3-Embedding-8B model, rounded to 4 decimal places.

Source

Items from the International Personality Item Pool (IPIP; <https://ipip.ori.org/>), which is in the public domain. Embeddings were generated with the Qwen/Qwen3-Embedding-8B model (<https://huggingface.co/Qwen/Qwen3-Embedding-8B>) and rounded to 4 decimal places to reduce file size. The regeneration script is in data-raw/big5.R.

Examples

```
data(big5)
str(big5)
table(big5$factors, big5$scoring)
```

plot.sfa_coverage	<i>Plot a Content-Validity Audit</i>
-------------------	--------------------------------------

Description

Plot a Content-Validity Audit

Usage

```
## S3 method for class 'sfa_coverage'
plot(x, type = c("coverage", "relevance", "curve"), seed = 20260711, ...)
```

Arguments

x	An "sfa_coverage" object from sfa_coverage() .
type	"coverage" (default) draws the proportional-overlap Euler diagram: two equal disks whose overlap area equals the measured construct coverage, filled with the real texts (dots) and items (triangles) placed by their full-space verdicts. "relevance" draws the per-item chart: corroboration counts, empirical p-values, and the calibrated critical count. "curve" draws the coverage curve against the matched-size null.
seed	Seed for the (arbitrary, uniform) within-region point placement in the "coverage" diagram. Default 20260711.
...	Passed to the type-specific plotter. For type = "relevance", wrap sets the character width at which item labels wrap (default 52; a large value such as 200 keeps every item on one line, and the left margin widens to fit).

Value

x, invisibly.

plot.sfa_coverage_battery	<i>Plot One Factor of a Content-Validity Audit Battery</i>
---------------------------	--

Description

Plot One Factor of a Content-Validity Audit Battery

Usage

```
## S3 method for class 'sfa_coverage_battery'
plot(x, factor = names(x)[1L], ...)
```

Arguments

<code>x</code>	An "sfa_coverage_battery" from <code>sfa_coverage()</code> on a multi-factor scale.
<code>factor</code>	Which factor's audit to plot. Default: the first.
<code>...</code>	Passed to <code>plot.sfa_coverage()</code> (e.g. <code>type = "relevance"</code>).

Value

The plotted "sfa_coverage" audit, invisibly.

sfa	<i>Semantic Factor Analysis</i>
-----	---------------------------------

Description

Performs exploratory factor analysis on language model embeddings of scale items. Given item text, sfa embeds each item, transforms embeddings into a similarity matrix, and runs EFA to recover latent factor structure entirely from the text.

Usage

```
sfa(
  items,
  nfactors = NULL,
  rotate = "oblimin",
  fm = "minres",
  encoding = "atomic",
  embed = "sbert",
  model = NULL,
  embeddings = NULL,
  similarity = NULL,
  scoring = NULL,
  n_factors_method = "parallel",
  dim_select = c("none", "dynega"),
  n.obs = NA,
  parallel_iter = 100L,
  seed = 42L,
  calibrate = FALSE,
  calibrate_iter = 100L,
  label_factors = FALSE,
  leximax = list(),
  ...
)
```

Arguments

items	Character vector of item text, or a data.frame with an item (or text) column and optional code, factor, scoring columns.
nfactors	Integer number of factors to extract, or NULL for automatic determination via <code>n_factors_method</code> .
rotate	Rotation method passed to fa . Default "oblimin" (requires GPArotation , which is in Imports).
fm	Extraction method passed to fa . Default "minres".
encoding	Similarity transform: "atomic" (default), "atomic_reversed", "squid", or "mean_centered_pearson". Use "atomic_reversed" with a scoring vector to sign-flip reverse-keyed items. See sfa_similarity .
embed	Embedding backend: "sbert", "openai", or a function. Ignored when embeddings is provided.
model	Model name for the embedding backend. If NULL (default), resolves to a backend-appropriate default: "Qwen/Qwen3-Embedding-0.6B" (about 1.2 GB) for "sbert" and "text-embedding-3-small" for "openai". The sbert default is chosen to run on any machine. Larger embedding models recover factor structure more accurately; for higher fidelity pass "Qwen/Qwen3-Embedding-4B" (about 8 GB RAM) or "Qwen/Qwen3-Embedding-8B" (about 16 GB RAM). When the default model is used, <code>print()</code> reminds you of these options.
embeddings	Optional precomputed numeric matrix (n_items x embedding_dim). When supplied, skips the embedding step entirely.
similarity	Optional precomputed symmetric item-by-item similarity matrix (n_items x n_items). When supplied, embedding and the encoding transform are skipped and this matrix is used directly — e.g. a signed NLI matrix from sfa_nli_matrix . Parallel analysis is unavailable in this mode (no embeddings), so retention falls back to "kaiser" unless nfactors is set.
scoring	Numeric vector of +1/-1 per item. If NULL, defaults to all +1 with an informative message for encoding methods that use it.
n_factors_method	Retention rule when nfactors = NULL: "parallel" (embedding-adapted, default), "kaiser", "EGA", "TEFI", or "semk" (calibrated semantic retention via the learned sem-k rule; see sfa_semk() — requires Python and a one-time model download).
dim_select	Embedding-dimension selection before analysis: "none" (default, use the full vector) or "dynega" (select the leading-coordinate depth that best recovers structure by EGA-based depth optimization, adapting Golino 2026; see sfa_dimselect). Requires EGAnet .
n.obs	Sample size passed to fa . NA (default) suppresses sample-size-dependent fit indices.
parallel_iter	Iterations for embedding parallel analysis.
seed	Random seed for stochastic operations, used via with_seed without touching the global RNG state.

calibrate	Logical: run an isotropic random-embedding Monte Carlo null calibration of the fit diagnostics? (Inspired by Pokropek 2026, but using a random-Gaussian unit-vector null rather than Pokropek’s corpus-word resampling. The two nulls differ in kind: corpus resampling preserves the baseline thematic similarity that all words in a topic-specific corpus share, whereas the Gaussian unit-vector null has zero expected inter-item similarity and is therefore a stricter, structure-free reference.)
calibrate_iter	Iterations for calibration.
label_factors	If TRUE, run <code>sfa_name()</code> on the fitted object with default settings and store the result as <code>\$labels</code> . Requires the candidate pool for the embedding model (fetched on first use; see <code>sfa_pool()</code>). Default FALSE.
leximax	Options list for <code>rotate = "leximax"</code> , ignored otherwise. Recognized entries: <code>lexmap</code> (a precomputed <code>sfa_lexmap()</code> object; built automatically when absent), <code>model</code> , <code>instruction</code> , <code>pool</code> , and <code>block_size</code> (forwarded to <code>sfa_lexmap()</code>), plus <code>n_random</code> , <code>seed</code> , <code>col_scale</code> , <code>rotation</code> , <code>normalize</code> , and <code>max_iter</code> (forwarded to <code>sfa_leximax()</code>).
...	Additional arguments passed to <code>fa</code> .

Value

An object of class "sfa" containing factor loadings, communalities, eigenvalues, variance accounted for, and embedding-specific diagnostics (KMO, TEFI, RMSR, CAF, McDonald’s omega). The `$loadings` component has class "loadings" and works with `factor.congruence` and `fa.sort`. Use `as_psych` to obtain the underlying `psych::fa` object.

References

- Milano, N., Luongo, M., Ponticorvo, M., & Marocco, D. (2025). Semantic analysis of test items through large language model embeddings predicts a-priori factorial structure of personality tests. *Current Research in Behavioral Sciences*, 8, 100168. doi:10.1016/j.crbeha.2025.100168
- Casella, M., Luongo, M., Marocco, D., Milano, N., & Ponticorvo, M. (2024). LLM embeddings on test items predict post hoc loadings in personality tests. *Ital-IA 2024: 4th National Conference on Artificial Intelligence*, CEUR Workshop Proceedings.
- Guenole, N., D’Urso, E. D., Samo, A., Sun, T., & Haslbeck, J. M. B. (Preprint). Enhancing Scale Development: Pseudo Factor Analysis of Language Embedding Similarity Matrices. OSF. <https://osf.io/3mpzb/>
- Pellert, M., Lechner, C. M., Sen, I., & Strohmaier, M. (2026). Neural network embeddings recover value dimensions from psychometric survey items on par with human data. *Findings of the Association for Computational Linguistics: EACL 2026*, 5738–5752.
- Pokropek, A. (2026). From keyword-based text measures to latent variables: Confirmatory factor analysis with word embeddings. *EPJ Data Science*. doi:10.1140/epjds/s13688026006541

See Also

`sfa_similarity`, `sfa_parallel`, `sfa_nfactors`, `sfa_embed`, `sfa_congruence`, `as_psych`

Examples

```
data(big5)
# nfactors = 5 keeps this example fast; omit it to let embedding-adapted
# parallel analysis (sfa_parallel) choose the number of factors.
fit <- sfa(big5$items, embeddings = big5$embeddings, scoring = big5$scoring,
          nfactors = 5)
print(fit)
plot(fit, type = "scree")
```

sfa_anchor

Construct-Label and Centroid Anchoring

Description

Produces an item-by-construct similarity matrix — the embedding analogue of a factor-loading table. Similarities are computed in the raw, un-flipped embedding space: embeddings encode topic, not valence, so a reverse-keyed item is still topically close to its construct and no sign-alignment is applied. Each cell is a *belonging strength*: high means the item belongs to that construct (for forward and reverse items alike), low means it does not. Read it like a loadings matrix — a well-behaved item is high in its own construct's column and low in the others; an item whose largest value lands on a different construct is a semantic cross-loader and a candidate for review.

Usage

```
sfa_anchor(
  x,
  anchor = c("centroid", "label", "both"),
  labels = NULL,
  label_embeddings = NULL,
  embed = NULL,
  model = NULL
)
```

Arguments

x	An object of class "sfa" carrying theoretical factor labels (i.e. fit from items with a factor column).
anchor	One of "centroid" (default), "label", or "both".
labels	Optional construct labels for the label anchor: either a character vector (one per construct, in the order of <code>unique(factors)</code>) or a named vector mapping construct -> label text. Defaults to the construct names themselves.
label_embeddings	Optional precomputed numeric matrix of label embeddings (one row per construct; named rows are matched to constructs). Use when the sfa object was built from precomputed embeddings.
embed, model	Embedding backend and model for the label anchor. Default to the backend/model recorded on x.

Details

Two anchor types are available:

"centroid" (default) Each construct's anchor is the mean of its own (un-flipped) item embeddings. An item's similarity to its own construct is computed leave-one-out (the item is excluded from its own anchor), mirroring a corrected item-total correlation. Self-contained — needs no construct text and works for any sfa object.

"label" Each construct's anchor is the embedding of the construct's name (or a richer gloss supplied via labels). Requires an embedding backend or precomputed label_embeddings. Because it uses the raw item embeddings, it is independent of the encoding the fit happened to use.

Value

An object of class "sfa_anchor": a list with the requested centroid and/or label item-by-construct similarity matrices, plus constructs, factors, and codes.

References

Wulff, D. U., & Mata, R. (2025). Semantic embeddings reveal and address taxonomic incommensurability in psychological measurement. *Nature Human Behaviour*, 9(5), 944–954. doi:10.1038/s4156202402089y

See Also

[sfa_simplify](#), [sfa](#)

Examples

```
data(big5)
fit <- sfa(
  data.frame(code = big5$codes, item = big5$items,
    factor = big5$factors, scoring = big5$scoring),
  embeddings = big5$embeddings, scoring = big5$scoring, nfactors = 5)

# item-by-construct belonging matrix (read like a loadings table)
a <- sfa_anchor(fit, anchor = "centroid")
head(round(a$centroid, 2))
```

sfa_build_bank

Pre-Embed a Set of Texts into a Reusable Bank

Description

Embeds texts (typically: every scale item, every construct definition, and every planned narrowing of a study) under one encoder and stores the vectors keyed by the exact strings [sfa_coverage\(\)](#) will pass to its embedder. Feed the result to audits via [sfa_embedding_bank\(\)](#).

Usage

```
sfa_build_bank(
  texts,
  instruction = TRUE,
  embed = "sbert",
  model = NULL,
  cache = TRUE,
  file = NULL
)
```

Arguments

texts	Character vector of raw texts (items, definitions). Wrapped with instruction exactly as sfa_coverage() wraps them, so lookups match.
instruction	TRUE (the construct-retrieval instruction the regions use, default), FALSE, or a custom string. Must match the regions the bank will be used with.
embed, model, cache	As in sfa_embed() .
file	Optional path to save the bank (.rds).

Value

An "sfa_bank": vectors, texts, instruction, encoder, timestamp.

sfa_build_region	<i>Build a Construct Region from a Text Corpus</i>
------------------	--

Description

Assembles the corpus half of a content-validity audit: a *construct region*, the cloud of real sentences that mention a construct term, embedded in the same space that [sfa_coverage\(\)](#) will embed the scale's items into. The result is a self-contained object recording its own provenance (corpus, extraction parameters, encoder, date); save it with `file =` and archive it with your analysis so the audit is reproducible.

Usage

```
sfa_build_region(
  construct,
  definition,
  corpus = "fineweb-10bt",
  target = 1500,
  max_docs = 2e+07,
  sentences_per_doc = 3,
  min_chars = 30,
  max_chars = 500,
```

```

    variants = NULL,
    embed = "sbert",
    model = NULL,
    instruction = TRUE,
    cache = TRUE,
    file = NULL,
    progress = TRUE
)

```

Arguments

construct	Construct term to search for, e.g. "procrastination".
definition	One- or two-sentence definition of the intended sense. Stored with the region and used as the default sense-gate seed in sfa_coverage() .
corpus	"fineweb-10bt" (default; streams from the Hugging Face Hub), a character vector of documents, a data frame with a text/item column, or a character vector of file or directory paths.
target	Stop once this many matching sentences are collected. Default 1500.
max_docs	Maximum number of corpus documents to scan. Default 2e7 (covers the full FineWeb 10BT sample). Lower it to cap runtime on a laptop; the saturation diagnostics in sfa_coverage() show whether the smaller region was enough.
sentences_per_doc	Maximum sentences kept per document (guards against one document flooding the region). Default 3.
min_chars, max_chars	Sentence length bounds. Defaults 30 and 500.
variants	Character vector of term spellings to match. Default NULL generates simple morphological variants of construct.
embed, model, cache	Passed to sfa_embed() ; model defaults to the package's default encoder. The audit must use the same encoder, which sfa_coverage() enforces from the region's metadata.
instruction	TRUE (default) embeds sentences under the construct-retrieval instruction (recommended: this register alignment outperformed alternatives in validation), FALSE embeds raw text, or a custom instruction string.
file	Optional path; when given, the region is saved there with saveRDS() and can be reloaded with sfa_load_region() .
progress	Print progress while streaming. Default TRUE.

Details

Two kinds of corpus are supported. `corpus = "fineweb-10bt"` streams the sample-10BT configuration of the FineWeb corpus (a documented random sample of a modern LLM-training corpus) from the Hugging Face Hub via the Python datasets package, stopping as soon as `target` sentences are found or `max_docs` documents have been scanned; common construct terms hit their quota within minutes, rare terms scan the full sample (use an HPC batch job, or lower `max_docs`

and check saturation). Alternatively, pass your own corpus: a character vector of documents, a data frame with a `text` column, or paths to plain-text files or directories. A domain-specific corpus (for example, workplace communications for a workplace construct) is a fully disclosed design choice recorded in the region's provenance.

Sentences are *not* sense-filtered here: the region stores every mention, and `sfa_coverage()` applies its sense gate at audit time against the definition supplied there. This is what makes construct narrowing cheap: one region file for "procrastination" can be re-audited as "academic procrastination" by re-gating with a narrower definition, with no new extraction.

Value

An object of class "sfa_region": a list with the sentences and their sources, the embedding matrix, and full provenance metadata.

See Also

`sfa_coverage()` to audit a scale against the region, `sfa_load_region()` to reload a saved region.

Examples

```
## Not run:
region <- sfa_build_region(
  construct = "procrastination",
  definition = paste("Procrastination is the voluntary delay of an",
                    "intended action despite expecting to be worse off."),
  file      = "procrastination_region.rds"
)

# a laptop-friendly build capped at 2M documents
region <- sfa_build_region("procrastination", definition = "...",
                          max_docs = 2e6)

# your own corpus
region <- sfa_build_region("procrastination", definition = "...",
                          corpus = "~/corpora/workplace_emails/")

## End(Not run)
```

sfa_build_regions

Build Many Construct Regions in One Corpus Pass

Description

The campaign-scale companion to `sfa_build_region()`: streams the corpus once and extracts sentences for every construct simultaneously, then builds one "sfa_region" per construct. Variants are matched on **word boundaries** (a component like "care" must not match "career"), which also makes compositional names safe: a construct like Honesty-Humility is gathered through variants = c("honesty", "humility").

Usage

```
sfa_build_regions(
  constructs,
  corpus = "fineweb-10bt",
  target = 1500,
  max_docs = 2e+07,
  sentences_per_doc = 3,
  min_chars = 30,
  max_chars = 500,
  embeddings = TRUE,
  embed = "sbert",
  model = NULL,
  cache = TRUE,
  instruction = TRUE,
  dir = NULL,
  progress = TRUE
)
```

Arguments

constructs	Named list: construct name -> list(definition = , variants = NULL) (a bare definition string also works). Default variants are simple inflections of the name.
corpus	"fineweb-10bt" (streamed once for all constructs), or a local corpus as in sfa_build_region() .
target, max_docs, sentences_per_doc, min_chars, max_chars	As in sfa_build_region() , applied per construct.
embeddings	Embed each region now? FALSE builds sentence-only regions (no encoder needed; sfa_coverage() refuses them until sfa_reembed_region() fills the embeddings in).
embed, model, cache, instruction	As in sfa_build_region() .
dir	Optional directory: each region is saved as {dir}/{slug}.rds.
progress	Print streaming progress? Default TRUE.

Details

For encoder-ladder studies, build once with embeddings = FALSE (a pure extraction; no encoder touched) and embed the same sentence sets under each encoder with [sfa_reembed_region\(\)](#) - the regions differ only in the embedding space, never in their text.

Value

A named list of "sfa_region" objects.

Description

Computes a comparison-data misfit profile for an embedding similarity structure, adapting the comparison data method of Ruscio and Roche (2012) to the response-free setting. For each candidate factor count k , the function builds a finite population of comparison data with known k -factor structure that reproduces both the model-implied correlation matrix and the empirical marginal distributions (an iterative rank-remapping refinement after the GenData program of Ruscio & Kaczetow, 2008), draws bootstrap samples of the empirical size, and records how well each sample's eigenvalue profile reproduces the observed one (root-mean-square residual, RMSR).

Usage

```
sfa_cd(
  x,
  input = c("embeddings", "data"),
  n_factors_max = 10L,
  n_samples = 500L,
  n_pop = 10000L,
  alpha = NULL,
  fm = "minres",
  gen_iter = 6L,
  seed = 42L
)
```

Arguments

<code>x</code>	Item embeddings ($n_items \times embedding_dim$; also accepts a fitted "sfa" object or an "sfa_embeddings" object from sfa_load_npz()) when <code>input = "embeddings"</code> , or a raw data matrix (cases $\times$ variables, e.g. survey responses) when <code>input = "data"</code> .
<code>input</code>	Whether <code>x</code> holds item embeddings (default) or raw case-by-variable data.
<code>n_factors_max</code>	Largest factor count to profile (default 10, capped at $\text{floor}(\text{variables} / 3)$).
<code>n_samples</code>	Bootstrap samples per factor count (default 500).
<code>n_pop</code>	Size of each comparison population (default 10000).
<code>alpha</code>	Optional alpha for Ruscio and Roche's sequential Mann-Whitney stopping rule. Default NULL skips the rule; see Details for why.
<code>fm</code>	Factor extraction method for the comparison models (default "minres").
<code>gen_iter</code>	Refinement iterations for the population generator (default 6; continuous marginals converge in a few refinements).
<code>seed</code>	Random seed, used via withr::with_seed() without touching the global RNG state.

Details

The deliverable is the profile, not a verdict. On conventional response data with a crisp factor boundary, the profile shows a sharp elbow at the true count. On embedding similarity matrices the misfit typically declines smoothly without an elbow, because a k-factor model with diagonal uniqueness cannot reproduce the heavy anisotropic tail of an embedding spectrum, and each added factor keeps improving reproduction. For the same reason Ruscio and Roche's sequential significance rule (each k tested against k - 1 with a one-tailed Mann-Whitney test) saturates at n_factors_max on embedding matrices at any conventional alpha, and in this package's benchmark runs it inflated with the case count on response data too. The rule is therefore only run when alpha is supplied explicitly, and its verdict should be read alongside the profile shape rather than in place of it.

With input = "embeddings", cases are embedding dimensions: the Pearson correlations of the transposed embedding matrix equal the "mean_centered_pearson" similarity of `sfa_similarity()`, so the profile addresses exactly the matrix that encoding factors. Other encodings are not correlation matrices of any data matrix, so the profile is computed in the correlation metric regardless.

Value

A list of class "sfa_cd" with components:

median_rmsr Numeric vector: median RMSR at each factor count (NA where extraction failed).

profile Numeric vector: median RMSR normalized by its one-factor value.

improvement Numeric vector: relative improvement (proportion) from each factor count to the next.

rmsr Numeric matrix (n_samples x n_factors_max): the full RMSR distributions.

eigenvalues Numeric vector: observed eigenvalues (descending).

n_factors Integer: sequential-rule verdict, only when alpha was supplied (otherwise NA).

alpha, n, n_samples, n_pop Settings used.

References

Ruscio, J., & Roche, B. (2012). Determining the number of factors to retain in an exploratory factor analysis using comparison data of known factorial structure. *Psychological Assessment*, 24(2), 282–292. doi:10.1037/a0025697

Ruscio, J., & Kaczetow, W. (2008). Simulating multivariate nonnormal data using an iterative algorithm. *Multivariate Behavioral Research*, 43(3), 355–381. doi:10.1080/00273170802285693

Goretzko, D., & Ruscio, J. (2024). The comparison data forest: A new comparison data approach to determine the number of factors in exploratory factor analysis. *Behavior Research Methods*, 56, 1838–1851. doi:10.3758/s13428023021224

Examples

```
## Not run:
data(big5)
cd <- sfa_cd(big5$embeddings, n_samples = 100)
print(cd)
plot(cd)
```

End(Not run)

sfa_clear_cache	<i>Clear Embedding Cache</i>
-----------------	------------------------------

Description

Removes all cached embedding files created by `sfa_embed()`.

Usage

`sfa_clear_cache()`

Value

Invisible NULL.

sfa_combine_banks	<i>Combine Embedding Lookups</i>
-------------------	----------------------------------

Description

Chains embed-functions (from `sfa_embedding_bank()` or `sfa_region_bank()`) into one: each text is served by the first lookup that contains it. Needed when one audit embeds texts from two sources, e.g. region-drawn pretend items (a region bank) plus the target region's definition (the main bank).

Usage

`sfa_combine_banks(...)`

Arguments

`...` Embed functions, tried in order.

Value

A function (texts) -> matrix, for use as embed = .

sfa_congruence	<i>Compare Semantic and Empirical Factor Structures</i>
----------------	---

Description

Computes agreement metrics between a semantic factor analysis result and a reference factor structure (from empirical data or theory).

Usage

```
sfa_congruence(
  sfa_fit,
  target,
  metrics = c("tucker", "nmi", "ari", "frobenius", "disattenuated")
)
```

Arguments

sfa_fit	An object of class "sfa".
target	A <code>psych::fa</code> object, a loadings matrix, a named factor label vector (one per item), or a correlation/similarity matrix.
metrics	Character vector of metrics to compute. Supported: "tucker", "nmi", "ari", "frobenius", "disattenuated".

Details

The disattenuated metric applies Spearman's (1904) correction $r / \sqrt{r_{xx'} r_{yy'}}$ to the correlation between the two matrices' item-pair values, estimating each matrix's reliability by a Spearman-Brown-corrected odd/even split-half of its lower-triangle entries. That reliability construction is this package's own device (values are capped at 1). When either split-half reliability is not positive — as the checkerboard sign pattern of "atomic_reversed" produces — the correction is undefined and NA is returned with a warning.

Value

A list of class "sfa_congruence" with one component per requested metric.

References

- Hubert, L., & Arabie, P. (1985). Comparing partitions (adjusted Rand index). *Journal of Classification*, 2, 193–218. doi:10.1007/BF01908075
- Strehl, A., & Ghosh, J. (2002). Cluster ensembles — a knowledge reuse framework for combining multiple partitions (geometric-mean normalized mutual information). *Journal of Machine Learning Research*, 3, 583–617.
- Spearman, C. (1904). The proof and measurement of association between two things (disattenuation for unreliability). *The American Journal of Psychology*, 15(1), 72–101. doi:10.2307/1412159

sfa_corplot

*Heatmap of an Item-by-Item Similarity Matrix***Description**

Draws a [cor.plot](#) heatmap of a semantic similarity matrix with sensible defaults for a many-item scale. By default the items are **grouped by their subscale/factor** (so each construct forms a block on the diagonal), the bulky transformed-embeddings attribute is removed, and all axis labels are shown.

Usage

```
sfa_corplot(
  x,
  factors = NULL,
  labels = NULL,
  group = TRUE,
  order = NULL,
  numbers = FALSE,
  upper = TRUE,
  gap.axis = -1,
  cex.axis = 0.75,
  xlas = 2,
  ...
)
```

Arguments

x	An "sfa" object, or a similarity matrix from sfa_similarity .
factors	Optional per-item subscale labels used to group the items. For an "sfa" object, defaults to its theoretical factors; for a matrix, defaults to a "factors" attribute if present.
labels	Optional per-item axis labels. By default uses short item codes (the code column for an "sfa" object, or a "codes" attribute / short dimnames on a matrix). If only sentence-like labels are available, compact codes are generated from the factors (e.g. A1, A2, D1, ...) rather than printing full item text.
group	Logical: reorder items so each factor forms a contiguous block (default TRUE). Ignored when no factors are available.
order	Optional character vector giving the order of the factor blocks (default: alphabetical). Entries are matched to the factor labels by exact, case-insensitive, or unique-prefix match, so for Depression/Anxiety/Stress both c("Depression", "Anxiety", "Stress") and c("D", "A", "S") work. A non-matching or ambiguous entry is an error; any factors omitted from order are appended after the listed ones.
numbers, upper, gap.axis, cex.axis, xlas	Passed to cor.plot ; defaults are tuned for a many-item matrix (no in-cell numbers, upper triangle, every label shown, small label text).
...	Further arguments passed to cor.plot .

Details

Grouping happens only for display — the underlying similarity matrix from [sfa_similarity](#) keeps its original item order (rows aligned with the items' scoring, codes, and embeddings), which the rest of the package relies on.

Value

The (grouped, relabelled) matrix that was plotted, invisibly.

See Also

[sfa_similarity](#), [sfa](#)

Examples

```
data(big5)
fit <- sfa(
  data.frame(code = big5$codes, item = big5$items,
             factor = big5$factors, scoring = big5$scoring),
  embeddings = big5$embeddings, scoring = big5$scoring, nfactors = 5)

sfa_corplot(fit)                                # heatmap, grouped by the Big Five
```

sfa_coverage

Audit the Content Validity of a Scale Against a Construct Region

Description

Quantifies content validity as geometric overlap between a scale's items and a construct region built with [sfa_build_region\(\)](#). Two headline numbers decompose the audit into Messick's two threats, each named by what it scores: **construct coverage** (the fraction of construct texts with an item within the coverage radius; its complement is construct *underrepresentation*) and **item relevance** (the fraction of items passing a per-item test against the ideal-item null; its complement is construct-*irrelevant* variance).

Usage

```
sfa_coverage(
  items,
  region,
  definition = NULL,
  keep_frac = NULL,
  construct = NULL,
  factor = NULL,
  cross = FALSE,
  radius_q = 0.95,
  radius = NULL,
  alpha = 0.05,
```

```

p_adjust = c("none", "BH"),
n_draws = 20L,
n_null = 200L,
sense_gate = TRUE,
min_silhouette = 0.65,
trim = 0.25,
screen_items = TRUE,
overlap_threshold = 0.6,
n_boot = 200L,
max_gaps = 6L,
gap_quotes = 3L,
embed = "sbert",
model = NULL,
cache = TRUE,
seed = 1L,
delta_q = NULL,
k_precision = NULL
)

```

Arguments

items	Character vector of item text, or a data frame with an item/text column (and optionally code), as in sfa() . When the data frame also has a factor column (subscale assignments), the audit runs per factor by default - one audit per (item set, construct claim) pair, which is the unit content validity is defined on - and returns an "sfa_coverage_battery". Use factor to restrict to a subset.
region	An "sfa_region" object from sfa_build_region() , the path to one saved with its file argument, or - for multi-factor audits - a list of regions named by factor (each factor is audited against its own construct region). A single region with a multi-factor scale audits every factor against that same region, with a message.
definition	Optional definition overriding the region's stored one - the construct-narrowing workflow. When supplied, the region is first restricted to the sentences the narrower definition explains best, then the usual filters apply with the new definition as seed. How the restriction is made depends on keep_frac (see below). Default NULL uses region\$definition with no restriction.
keep_frac	How much of the region a supplied definition keeps. NULL (default) uses the comparative rule: keep the sentences more similar to the narrower definition than to the region's original one. That rule is strict - when the narrower definition is a close paraphrase of the original, sentences containing the construct term hug the original definition and the rule can keep almost nothing. A number in (0, 1] switches to rank narrowing : keep the keep_frac fraction of the region most similar to the narrower definition, which always yields a region of known size. Rank narrowing is the recommended workflow for testing a narrowed construct claim, usually together with a fixed radius from the un-narrowed audit (narrowing shrinks the region, and recalibrating on the shrunk region would change the yardstick along with the claim).

construct	Optional display label for the (possibly narrowed) construct. Default: the region's construct, or the first words of a supplied definition.
factor	Which factors to audit when the items carry factor assignments. Default NULL audits all of them. A character vector of factor names restricts the battery to that subset (a single name returns a plain "sfa_coverage" audit). With character-vector items lacking assignments, factor may instead be a vector of one assignment per item.
cross	Audit every factor against every region (requires factor assignments and a named region list)? Default FALSE. The result is an "sfa_coverage_cross" matrix of audits - the content analogue of a multitrait matrix: items should be relevant to their own construct's region (diagonal) and irrelevant to their siblings' (off-diagonal), which is discriminant content validity measured from item text alone. Off-diagonal relevance is floored by how separable the constructs are in language, not by zero - the printed output states this caveat. sfa_cross_matrix() extracts the numeric matrix.
radius_q	Null quantile defining the coverage radius. Default 0.95 (ideal scale covers ~95% of the region; each gap call is a test at $\alpha = 1 - \text{radius_q}$).
radius	Optional fixed coverage radius, overriding the calibrated one - the anchoring workflow, the analogue of holding a test's cut score fixed when equating forms. Calibration ties the radius to the audited condition, so any comparison that changes the item set or the region (deleting items, auditing a short form, rank narrowing) would otherwise move the yardstick along with the thing being measured: deleting items <i>grows</i> the calibrated radius and can raise coverage. For such comparisons, take <code>\$radius</code> from the reference audit and pass it here so every condition is scored against the same yardstick. The item-relevance null is drawn at the fixed radius, and bootstrap resamples hold it fixed rather than recalibrating. The calibrated radius is still computed and returned as <code>\$radius_calibrated</code> for comparison. Default NULL calibrates as usual.
alpha	Per-item test level against the ideal-item null. Default 0.05. An item is flagged when its corroboration count's empirical p-value is at most alpha.
p_adjust	Multiplicity handling for the item flags: "none" (per-item tests, default) or "BH" (Benjamini-Hochberg false discovery rate across the scale's items: of the flagged items, at most alpha are expected to be false alarms).
n_draws	Draws for the matched-size null behind the radius. Default 20.
n_null	Draws for the ideal-item count null behind the p-values (more draws give finer p resolution). Default 200.
sense_gate	Apply the sense gate (2-means on seed similarity; drops the wrong-sense/incidental-mention mode only when the split is real)? Default TRUE.
min_silhouette	Minimum 1-d silhouette for the sense gate to fire. Default 0.65 (a unimodal 1-d Gaussian scores ~0.55 under a forced 2-means split; genuine sense mixtures score higher).
trim	Incidental-mention trim: fraction of the region with the lowest similarity to the definition to drop before auditing (web corpora carry a tail of sentences that merely mention the term). Applied after the sense gate and reported in the print output. Default 0.25; set 0 to disable.

screen_items	Drop region sentences that near-duplicate the audited items (circularity rule)? Default TRUE.
overlap_threshold	Item-screen lexical threshold: a region sentence is dropped when at least this fraction of its content words appears in a single item (default 0.6), or - the self-calibrating geometric criterion, no constant - when it is more similar to an item than to any other region sentence (a leaked item copy always is; an on-topic corpus sentence never is; fixed cosine thresholds do not transfer across encoders' similarity scales).
n_boot	Bootstrap resamples of the region for percentile confidence intervals (0 to skip). Default 200.
max_gaps	Maximum gap clusters to report. Default 6.
gap_quotes	Example sentences quoted per gap. Default 3.
embed, model, cache	Passed to <code>sfa_embed()</code> for the items and seed. model defaults to the region's encoder; overriding it is almost always a mistake (the audit compares points in one space) and produces a warning.
seed	Random seed for the null draws, gap clustering, and bootstrap. Default 1.
delta_q, k_precision	Deprecated (pre-0.3.0 names). delta_q is mapped to radius_q with a warning. k_precision is ignored with a warning: the fixed-count rule it set was region-size dependent (sampling more construct text inflated relevance) and is replaced by the calibrated per-item test.

Details

One 95% convention calibrates both numbers against an **ideal same-length scale** - items drawn from the construct region itself. The coverage radius is the radius_q (default 95%) quantile of the matched-size null's nearest-neighbor distances, so an ideal scale's construct coverage is about 0.95 at any scale length, region size, or embedding dimension. Each item's **corroboration count** (construct texts within its radius) gets an empirical p-value against the ideal-item null; items are flagged at alpha (default .05), so an ideal scale's item relevance is also about 0.95. The identity behind the convention: $1 - \text{radius_q}$ and alpha are per-decision Type I error rates of Monte Carlo tests. Because the null's counts grow with region size, the critical count rescales automatically - sampling more construct text cannot inflate relevance.

The printed report frames the two remedies for low coverage explicitly: *add items* aimed at the named gaps (each gap is labeled with distinctive terms and quoted example sentences from the region - real corpus content the scale does not reach), or *narrow the construct claim* (the covered subregion's distinctive terms describe what the items actually measure). Narrowing is cheap to test: re-run `sfa_coverage()` on the same region with a narrower definition (for example "academic procrastination") and a `keep_frac`, which restricts the region at audit time - no new extraction. Hold radius at the un-narrowed audit's value so the narrowed claim is scored against the same yardstick (see the two parameters below).

Value

An object of class "sfa_coverage" with the audit numbers, per-item corroboration counts and p-values, filter accounting, gap report, and provenance - or, when the items carry factor assignments

and more than one factor is audited, an "sfa_coverage_battery": a named list of "sfa_coverage" audits, one per factor (x\$Depression is that factor's full audit). `print()` gives the report (a compact per-factor table for batteries); `plot()` draws the proportional-overlap diagram (type = "coverage"), the per-item relevance chart (type = "relevance"), or the coverage curve against the matched-size null (type = "curve"); `sfa_gaps()` returns the gap table.

Examples

```
## Not run:
region <- sfa_load_region("procrastination_region.rds")
audit <- sfa_coverage(my_items, region)
audit          # coverage, relevance, gaps, the two remedies
plot(audit)    # proportional-overlap (Euler) diagram
plot(audit, type = "relevance") # per-item counts, p-values, flags

# a multidimensional battery: items with a 'factor' column audit
# per subscale by default, each against its own construct region
battery <- sfa_coverage(dass_items,
  region = list(Depression = reg_dep,
                Anxiety    = reg_anx,
                Stress     = reg_str))

battery          # one row per factor
battery$Depression # full report for one subscale
sfa_coverage(dass_items, reg_dep, factor = "Depression") # just one

# test a narrower claim against the same region - no new extraction.
# Rank narrowing keeps the half of the region the narrower definition
# explains best; the fixed radius anchors the comparison to the full
# audit's calibration.
sfa_coverage(my_items, region, keep_frac = 0.5, radius = audit$radius,
  definition = "Academic procrastination is the delay of
    study-related tasks despite expecting costs.")

# the same anchoring applies to any item-set comparison, e.g. a
# deletion study: score the reduced scale at the full scale's radius
sfa_coverage(my_items[-drop_idx, ], region, radius = audit$radius)

## End(Not run)
```

sfa_cross_matrix

Extract the Numeric Matrix from a Cross-Audit

Description

Extract the Numeric Matrix from a Cross-Audit

Usage

```
sfa_cross_matrix(x, what = c("relevance", "coverage"))
```

Arguments

x	An "sfa_coverage_cross" from sfa_coverage(..., cross = TRUE).
what	"relevance" (default; the discriminant-informative number) or "coverage".

Value

A numeric matrix, factors in rows, regions in columns.

sfa_deletion_gaps	<i>Newly Uncovered Content After an Item-Set Change</i>
-------------------	---

Description

Compares two audits of the *same region* - a reference audit and an audit of a changed item set (items deleted, a short form, a revision) - and reports the content the change strands: the region sentences covered by the reference scale but not by the changed one. Labeling only this *delta* is what makes a deletion study readable. The changed scale's full gap report mixes newly created gaps with whatever the reference scale already failed to cover, and the pre-existing uncovered mass usually drowns the signal.

Usage

```
sfa_deletion_gaps(reference, reduced, quotes = 3L, top = 8L)
```

Arguments

reference	The reference audit ("sfa_coverage"), typically the full scale.
reduced	The changed item set's audit, run against the same region and anchored to reference\$radius.
quotes	Number of example sentences to return. Default 3, ordered by how far the change strands them (largest increase in nearest-item distance first).
top	Number of label terms. Default 8.

Details

Both audits must be run against the same region with the same filters, and the changed audit must be **anchored** to the reference audit's radius (radius = reference\$radius) - otherwise recalibration moves the yardstick along with the item set and the delta confounds the two. The function checks both conditions.

Value

A list: n_new (newly uncovered sentences), share_new (their fraction of the region), coverage_drop (reference\$coverage - reduced\$coverage), terms (distinctive terms of the newly uncovered content against the still-covered content), and quotes.

Description

Selects how many leading embedding coordinates ("depth") to use before factor analysis, instead of defaulting to the full vector. This **adapts the depth-optimization objective** of Golino (2026); it is *not* a reimplement of Dynamic EGA (DynEGA) – it does not perform DynEGA's time-delay embedding or derivative (GLLA) estimation, but applies static EGA at each depth and optimizes Golino's composite. Following Golino (2026), the embedding is treated as a searchable landscape: structural information is not uniformly distributed across coordinates, so a sub-range of dimensions can recover the construct structure more cleanly than the whole vector (and denoise the over-factoring seen with some embedding models).

Usage

```
sfa_dimselect(
  embeddings,
  factors = NULL,
  scoring = NULL,
  encoding = "atomic",
  min_depth = 3L,
  max_depth = NULL,
  step = NULL,
  max_eval = 150L,
  weights = c(nmi = 0.7, tefi = 0.3),
  algorithm = "walktrap"
)
```

Arguments

embeddings	Numeric matrix (n_items x embedding_dim).
factors	Optional character/factor vector of theoretical labels, one per item, enabling the NMI term. If NULL, TEFI-only selection.
scoring	Optional numeric +1/-1 vector (keying), passed to the similarity transform.
encoding	Similarity transform used at each depth (default "atomic", matching sfa). See sfa_similarity .
min_depth	Smallest depth to evaluate (default 3, with a minimum of 3 imposed for TMFG stability).
max_depth	Largest depth to evaluate (default: full embedding dimension).
step	Depth increment. Default chooses a step giving at most max_eval evaluations. (Golino 2026 swept depths in increments of 5 coordinates over a large range, not 5 total evaluations.)
max_eval	Soft cap on the number of depths evaluated when step is left at its default (default 150).

weights	Named numeric vector <code>c(nmi=, tefi=)</code> for the composite (default <code>c(nmi = 0.70, tefi = 0.30)</code>).
algorithm	Community-detection algorithm passed to EGAnet (default "walktrap").

Details

The coordinate index is swept as an ordered depth axis. The function sweeps increasing depths d ; at each depth it builds the item-by-item association matrix from the first d coordinates, estimates the network with the Triangulated Maximally Filtered Graph (TMFG) and detects communities with the Walktrap algorithm (both via **EGAnet**, as in Golino 2026), then scores the resulting partition with:

- the Total Entropy Fit Index (**TEFI**; lower is better), and
- Normalized Mutual Information (**NMI**) against the theoretical factor labels, when available (higher is better).

Both metrics are min-max normalized across the swept depths and combined into a composite $C(d) = w_{NMI} NMI_{norm} - w_{TEFI} TEFI_{norm}$ (default weights 0.70 / 0.30, per Golino 2026). The depth maximizing C is returned. With no theoretical labels the selection falls back to minimizing TEFI alone (less reliable; a single metric can yield structurally incoherent optima).

Value

An object of class "sfa_dimselect": a list with `optimal_depth`, the full trajectory data frame (`depth`, `n_dim`, `nmi`, `tefi`, and `normalized/composite` columns), the weights used, and `full_dim`.

Selection engine vs. analysis engine

Depth is scored with the EGA network / Walktrap partition (Golino's engine). When the chosen depth then feeds [fa](#)-based extraction (the default in [sfa](#)), the subspace that is best for EGA recovery is not guaranteed to be best for the EFA solution. For results that match the selection criterion, pair `dim_select = "dynega"` with `n_factors_method = "EGA"`. Golino (2026) also reports the largest gains for moderate-to-large item pools (roughly 10–20+ items per dimension); short scales may see little or no benefit.

References

Golino, H. (2026). Optimizing the landscape of LLM embeddings with Dynamic Exploratory Graph Analysis for generative psychometrics: A Monte Carlo study. Manuscript under review, *Proceedings of the 90th Annual International Meeting of the Psychometric Society*. arXiv:2601.17010.

See Also

[sfa](#) (use `dim_select = "dynega"`), [sfa_similarity](#)

Examples

```
data(big5)

if (requireNamespace("EGAnet", quietly = TRUE)) {
```

```
# small depth grid for a quick illustration
ds <- sfa_dimselect(big5$embeddings, factors = big5$factors,
                   scoring = big5$scoring, max_depth = 80, step = 20)
ds$optimal_depth
}
```

sfa_ekc

Empirical Kaiser Criterion for Embedding Similarity Matrices

Description

Applies the empirical Kaiser criterion (EKC; Braeken & van Assen, 2017) to an embedding similarity matrix, with the embedding dimension playing the role of the sample size.

Usage

```
sfa_ekc(sim_matrix, embeddings = NULL, n = NULL)
```

Arguments

<code>sim_matrix</code>	Numeric similarity matrix (<code>n_items</code> x <code>n_items</code>), or a fitted "sfa" object.
<code>embeddings</code>	Numeric embedding matrix (<code>n_items</code> x <code>embedding_dim</code>), used only for its column count. Optional if <code>n</code> is given.
<code>n</code>	Sample size to use in place of <code>ncol(embeddings)</code> : the embedding dimension for similarity matrices, or the respondent count when applying the criterion to a conventional correlation matrix.

Details

The EKC replaces Kaiser's fixed threshold of one with a series of reference eigenvalues. The first reference is the asymptotic maximum sample eigenvalue of a null-model correlation matrix, $(1 + \sqrt{\gamma})^2$ with γ the variables-to-sample-size ratio (Marchenko-Pastur upper edge). Each subsequent reference applies Braeken and van Assen's proportional correction for the variance already absorbed by preceding observed eigenvalues, floored at one. Retention counts the run of leading eigenvalues above their references (the paper's factors-1-to-K rule, operationally the same first-crossing stop as [sfa_parallel\(\)](#)). The serial correction addresses the classical parallel-analysis weakness that reference values ignore variance captured by real factors.

Adaptation note: with similarity matrices computed across embedding dimensions (see [sfa_similarity\(\)](#)), the sample size is the embedding dimension, so γ is items over dimensions. Embedding dimensions are coordinates rather than sampled respondents, so the Marchenko-Pastur bound is a heuristic reference here, not a sampling-theoretic one; treat the result as one voice among the criteria in [sfa_nfactors\(\)](#).

Value

A list of class "sfa_ekc" with components:

- n_factors** Integer: suggested number of factors.
- observed** Numeric vector: observed eigenvalues (descending).
- references** Numeric vector: EKC reference eigenvalues.
- n** The sample size used.

References

Braeken, J., & van Assen, M. A. L. M. (2017). An empirical Kaiser criterion. *Psychological Methods*, 22(3), 450–466. doi:10.1037/met0000074

Examples

```
data(big5)
sim <- sfa_similarity(big5$embeddings, "mean_centered_pearson")
sfa_ekc(sim, big5$embeddings)
```

sfa_embed	<i>Embed Item Text with a Language Model</i>
-----------	--

Description

Computes embeddings for a vector of item text using a sentence-transformer or other embedding backend.

Usage

```
sfa_embed(items, embed = "sbert", model = NULL, cache = TRUE, ...)
```

Arguments

items	Character vector of item text, or a data frame with an item/text column (and optionally a code column, used as rownames so short codes flow through to plots such as sfa_corplot).
embed	Embedding backend: "sbert" (default, via reticulate), "openai" (via httr2), or a function taking a character vector and returning a numeric matrix.
model	Model name passed to the backend. If NULL (default), a backend-appropriate default is used: "Qwen/Qwen3-Embedding-0.6B" for "sbert" and "text-embedding-3-small" for "openai". Larger embedding models recover factor structure more accurately; see sfa .
cache	Logical: cache embeddings in tools::R_user_dir("semanticfa", "cache")? Default TRUE.
...	Additional arguments passed to the embedding backend function.

Value

A numeric matrix (n_items x embedding_dim). Rownames are the item codes when items is a data frame with a code column, otherwise the item text.

sfa_embedding_bank	<i>Use a Pre-Generated Embedding Bank as an Embedder</i>
--------------------	--

Description

Turns an "sfa_bank" (or a path to one) into a lookup function that [sfa_coverage\(\)](#) and [sfa_embed\(\)](#) accept as their embed argument, so audits run entirely from published embeddings - no encoder loads.

Usage

```
sfa_embedding_bank(bank)
```

Arguments

bank	An "sfa_bank" from sfa_build_bank() , or a path to one.
------	---

Details

A lookup miss is an error, not a fallback: the bank must contain every item, definition, and planned narrowing the audit touches. The error lists what is missing so the bank can be rebuilt to include it.

Value

A function (texts) -> matrix, for use as embed = .

Examples

```
## Not run:
bank <- sfa_embedding_bank("bank_qwen8b.rds")
region <- sfa_load_region("regions_qwen8b/procrastination.rds")
audit <- sfa_coverage(my_items, region, embed = bank,
                     model = region$encoder)

## End(Not run)
```

sfa_gaps

*Gap Table from a Coverage Audit***Description**

Gap Table from a Coverage Audit

Usage

sfa_gaps(x)

Argumentsx An "sfa_coverage" object from [sfa_coverage\(\)](#).**Value**

A data frame with one row per gap: share of region mass, label terms, and an example quote.

sfa_install_python

*Provision the Python Environment for Embedding***Description**

Declares and installs the Python packages needed by the "sbert" embedding backend and the default [sfa_nli_matrix](#) classifier (sentence-transformers, which pulls in torch and transformers).

With **reticulate** (≥ 1.41) these requirements are also declared automatically on first use via `reticulate::py_require()`, so calling this is optional — it is handy for provisioning ahead of time (e.g. on a machine with internet before running offline) or into a specific environment.

Usage

sfa_install_python(packages = "sentence-transformers", ...)

Arguments

packages Character vector of Python packages to require/install.
 ... Passed to `reticulate::py_install()` (e.g. `envname`, `method`).

Value

Invisible NULL.

Examples

```
## Not run:
# one-time setup of the Python embedding environment
sfa_install_python()

## End(Not run)
```

sfa_itemplot	<i>2-D Item Map (t-SNE, UMAP, PCA, or MDS)</i>
--------------	--

Description

A 2-D scatter of the scale's items, the embedding-space companion to [sfa_corplot](#): each point is an item, points are coloured by their theoretical factor and labelled with their short code, so you can see at a glance which items cluster together, which sit between constructs, and which are outliers. Operates on the same (transformed) item embeddings the factor analysis uses, or on a similarity matrix (converted to a distance).

Usage

```
sfa_itemplot(
  x,
  method = c("tsne", "umap", "pca", "mds"),
  factors = NULL,
  labels = NULL,
  color = TRUE,
  perplexity = NULL,
  n_neighbors = NULL,
  seed = 42,
  pch = 19,
  cex = 0.9,
  legend = TRUE,
  ...
)

sfa_tsneplot(x, method = c("tsne", "umap", "pca", "mds"), ...)
```

Arguments

x	An "sfa" object (uses its item embeddings) or a symmetric numeric item-by-item similarity matrix.
method	Projection: "tsne" (default), "umap", "pca", or "mds" (classical multidimensional scaling). All four work out of the box (Rtsne and uwot are dependencies; PCA and MDS are base R). t-SNE and UMAP are better at showing local clusters but are only sensible above a handful of items.
factors, labels	Optional per-item factor labels and point labels (codes). Default to those carried on x (or the matrix's "factors"/"codes" attributes).

color	Logical; colour points by factor (default TRUE).
perplexity	t-SNE perplexity (method = "tsne"). If NULL, a safe value is chosen for the item count ($\max(1, \min(30, \text{floor}((n - 1) / 3)))$).
n_neighbors	UMAP neighbourhood size (method = "umap"). If NULL, $\min(15, n - 1)$.
seed	Random seed for reproducibility (t-SNE and UMAP are stochastic).
pch, cex	Point symbol and size.
legend	Logical; draw a factor legend (default TRUE).
...	Passed to plot .

Details

The projection method is selectable via method; method = "tsne" reproduces the original behaviour. `sfa_tsneplot()` is a deprecated alias kept for back-compatibility.

Value

Invisibly, a list with the 2-D coordinates Y, the factors, the labels, and the method used.

References

- van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.
- McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for dimension reduction. arXiv:1802.03426.

See Also

[sfa_corplot](#)

Examples

```
data(big5)
fit <- sfa(
  data.frame(code = big5$codes, item = big5$items,
             factor = big5$factors, scoring = big5$scoring),
  embeddings = big5$embeddings, scoring = big5$scoring, nfactors = 5)
sfa_itemplot(fit, method = "pca") # runnable: bundled data, base-R PCA
## Not run:
sfa_itemplot(fit)                # t-SNE (default)
sfa_itemplot(fit, method = "umap") # UMAP

## End(Not run)
```

sfa_item_fit

*Vet a Candidate Scale Item Before Data Collection***Description**

Scores draft item text against an existing scale: how well it matches each construct, whether it discriminates (low cross-loading risk), how it compares to the construct's current items, and whether it duplicates one of them — entirely response-free. Each candidate is scored on two complementary axes per construct:

Similarity to name Cosine between the candidate and the embedding of the construct's name (e.g. "Depression"): does it sound like the construct?

Similarity to other items Cosine between the candidate and the centroid of the construct's existing items: does it look like the other items?

When the two disagree they are informative: high name + low items is a *gap-filler* (on-topic but covering new ground); low name + high items is *drift* (looks like the items but not the construct).

Usage

```
sfa_item_fit(
  x,
  item,
  construct = NULL,
  reverse_key = FALSE,
  redundancy_cutoff = 0.9,
  embed = NULL,
  model = NULL
)
```

Arguments

<code>x</code>	An object of class "sfa" carrying theoretical factor labels and stored (raw) embeddings.
<code>item</code>	Character vector of one or more candidate items to vet.
<code>construct</code>	Optional name of the construct you intend the item for (matched to the factor labels by exact, case-insensitive, or unique-prefix match). When supplied, the verdict is reported relative to that construct as well as the best-matching one.
<code>reverse_key</code>	Logical; set TRUE to sign-flip the candidate's embedding before comparison. Against the un-flipped reference space this negates the candidate's similarities (the anti-topic direction), so it is a diagnostic contrast, not an assignment mode. Default FALSE.
<code>redundancy_cutoff</code>	Similarity to the nearest existing item at or above which the candidate is flagged as a near-duplicate. Default 0.90.
<code>embed, model</code>	Embedding backend and model used to embed the candidate(s) and the construct names. Default to those recorded on <code>x</code> .

Details

All comparisons run in the raw, un-flipped embedding space, matching [sfa_anchor](#) and [sfa_simplify](#): reference items are never sign-flipped by their scoring key, so belonging reflects topic. (A sign-flipped embedding is an anti-topic vector, not a reverse-scored meaning, and flipping reference items depresses the item-similarity profile of constructs with many reverse-keyed items.)

Value

An object of class "sfa_item_fit": a list with `similarity_to_name` and `similarity_to_items` (candidate x construct matrices), a per-candidate summary data frame (best construct, the two similarities, second-best construct and gap, strength versus the average existing item, nearest item and its similarity, and a verdict), and the per-construct average existing-item similarity `avg_item_fit`.

References

Wulff, D. U., & Mata, R. (2025). Semantic embeddings reveal and address taxonomic incommensurability in psychological measurement. *Nature Human Behaviour*, 9(5), 944–954. doi:10.1038/s4156202402089y

See Also

[sfa_anchor](#), [sfa_redundancy](#)

Examples

```
## Not run:
# embeds the candidates live, so this needs the Python backend (or embed=)
data(big5)
fit <- sfa(data.frame(code = big5$codes, item = big5$items,
                      factor = big5$factors),
           embeddings = big5$embeddings, nfactors = 5)
sfa_item_fit(fit, "I make friends easily.",
             model = "Qwen/Qwen3-Embedding-8B")
sfa_item_fit(fit, c("I am the life of every party.",
                   "I rarely feel anxious or depressed."),
             model = "Qwen/Qwen3-Embedding-8B") # vet several at once

## End(Not run)
```

Description

Compares whole scales by the meaning of their items versus the meaning of their names to surface two classic measurement problems (Wulff & Mata, 2025, 2026): **jingle** (scales with similar *names* but dissimilar *content*) and **jangle** (scales with dissimilar names but similar content).

Usage

```
sfa_jinglejangle(
  scales,
  labels = NULL,
  embed = "sbert",
  model = NULL,
  flag = 0.2,
  item_embeddings = NULL,
  label_embeddings = NULL
)
```

Arguments

scales	A named list; each element is a character vector of the scale's item texts. The names are used as scale labels unless labels is given.
labels	Optional character vector of scale names (construct labels), one per scale, overriding the list names.
embed, model	Embedding backend and model (default the package default sbert model).
flag	Absolute content-minus-label similarity difference at which to flag a pair (default 0.20). The single-difference rule and its 0.20 default are this package's convenience heuristic, not Wulff & Mata's criterion (they flag pairs with quantile-derived dual cutoffs on the two similarities separately); tune it to your scale set.
item_embeddings, label_embeddings	Optional precomputed embeddings: a named list of per-scale item-embedding matrices, and a matrix of label embeddings (one row per scale). Use when no embedding backend is available.

Details

Each scale is represented by a content vector (the mean of its item embeddings) and a label vector (the embedding of its name). For every pair of scales the function compares content similarity with label similarity; large divergences flag the two fallacies.

Value

An object of class "sfa_jinglejangle": a list with the content_sim and label_sim scale-by-scale matrices and a flags data frame (scale_a, scale_b, content_sim, label_sim, divergence, type).

References

- Wulff, D. U., & Mata, R. (2025). Semantic embeddings reveal and address taxonomic incommensurability in psychological measurement. *Nature Human Behaviour*, 9(5), 944–954. doi:10.1038/s4156202402089y
- Wulff, D. U., & Mata, R. (2026). Escaping the jingle-jangle jungle: Increasing conceptual clarity in psychology using large language models. *Current Directions in Psychological Science*, 35(2), 59–65. doi:10.1177/09637214251382083

See Also[sfa_anchor](#)**Examples**

```

data(big5)
scales <- list(
  Extraversion = big5$items[big5$factors == "Extraversion"],
  Sociability  = big5$items[big5$factors == "Extraversion"], # same content, new name
  Neuroticism  = big5$items[big5$factors == "Neuroticism"])

# precomputed embeddings so the example needs no backend
ie <- lapply(scales, function(items)
  big5$embeddings[match(items, big5$items), , drop = FALSE])
le <- big5$embeddings[match(c("E1", "C31", "N11"), big5$codes), , drop = FALSE]
sfa_jinglejangle(scales, item_embeddings = ie, label_embeddings = le)

## Not run:
# with a live backend, pass the scales and their names are embedded directly:
sfa_jinglejangle(scales)

## End(Not run)

```

sfa_leximax

*Leximax: Rotate Factor Axes Toward the Construct Lexicon***Description**

Chooses, among the orientations that reproduce a fitted factor solution identically, the one whose factors sit closest to real construct terms. The optimizer alternates naming (the full [sfa_name\(\)](#) selection rule on the current pattern matrix) with oblique target rotation toward the retrieved terms' predicted loading profiles, from multiple deterministic starts, until the retrieved term tuple recurs. The recurrent states are the candidate solutions (a fixed point is the one-state case), and the recurrent state with the highest nameability wins, with ties broken by start order and then iteration. The winner is re-named through the canonical blocked pool path and those labels are authoritative.

Usage

```

sfa_leximax(
  x,
  Phi = NULL,
  lexmap = NULL,
  model = NULL,
  pool = NULL,
  instruction = NULL,
  n_random = 10L,
  seed = 42L,
  col_scale = c("unitmax", "z"),

```

```

rotation = c("oblique", "orthogonal"),
normalize = FALSE,
max_iter = 30L,
block_size = 50000L,
baseline = NULL,
baseline_sd = NULL,
...
)

```

Arguments

<code>x</code>	An <code>sfa()</code> fit, a <code>psych::fa</code> fit, or a pattern matrix.
<code>Phi</code>	Factor correlations when <code>x</code> is a plain matrix (default identity).
<code>lexmap</code>	An <code>sfa_lexmap()</code> for the instrument's items. Required when <code>x</code> is a matrix; built automatically from an <code>sfa</code> fit.
<code>model, pool, instruction</code>	Passed to <code>sfa_lexmap()</code> when it must be built.
<code>n_random</code>	Number of seeded random orthonormal starts appended to the oblimin, varimax, and geominQ starts (default 10).
<code>seed</code>	Base seed for the random starts (default 42).
<code>col_scale</code>	Target column scaling: "unitmax" (default) or "z".
<code>rotation</code>	"oblique" (default) or "orthogonal".
<code>normalize</code>	Kaiser row normalization inside the target rotation.
<code>max_iter</code>	Iteration cap per start (default 30).
<code>block_size</code>	Passed to <code>sfa_lexmap()</code> when it must be built.
<code>baseline, baseline_sd</code>	Optional per-pool-word null mean and standard deviation for surprise scoring, passed through to <code>sfa_nameability()</code> . Default NULL scores raw target cosines.
<code>...</code>	Passed to the embedding backend via <code>sfa_lexmap()</code> .

Details

Also available as `rotate = "leximax"` in `sfa()`.

Value

An object of class `sfa_leximax`: loadings, `Phi`, `Th`, `A0`, labels (with the canonical retrieved labels and leave-one-out candidate sets), `criterion`, `per_factor`, `start`, `iteration`, `history`, `converged`.

sfa_lexmap

*Build the Lexical Map for Leximax Rotation***Description**

Precomputes everything leximax needs to score orientations of one instrument: the instruction-conditioned item embeddings, their Gram matrix, and the pool-by-item cosine matrix *S*. Building the map costs one pass over the candidate pool; every subsequent naming of any orientation of the same items is then nearly instant, which is what makes multi-start optimization and Monte Carlo calibration practical.

Usage

```
sfa_lexmap(
  x,
  model = NULL,
  instruction = NULL,
  pool = NULL,
  block_size = 50000L,
  ...
)
```

Arguments

<code>x</code>	An sfa() fit, or a character vector of item texts.
<code>model</code>	Naming model id. Default NULL follows sfa_name() : the fit's embedding model (or the package default) is used.
<code>instruction</code>	Naming instruction override (see sfa_naming_instruction()).
<code>pool</code>	An sfa_pool() object, or NULL to fetch it.
<code>block_size</code>	Pool rows per block while building <i>S</i> (memory knob; results are identical for any value).
<code>...</code>	Passed to the embedding backend.

Value

An object of class `sfa_lexmap`.

sfa_load_npz*Load Pre-generated Embeddings from a NumPy .npz File*

Description

Reads a NumPy .npz archive of pre-computed item embeddings (and, if present, the item codes, factor labels, scoring, and item text) into a tidy object that [sfa](#), [sfa_similarity](#), and [sfa_corplot](#) accept directly — so loading saved embeddings is one line instead of hand-rolling **reticulate**/NumPy calls.

Usage

```
sfa_load_npz(  
  path,  
  embeddings_key = "embeddings",  
  codes_key = "codes",  
  items_key = "items",  
  factors_key = "factors",  
  scoring_key = "scoring"  
)
```

Arguments

path	Path to a .npz file.
embeddings_key	Name of the embeddings array in the archive (default "embeddings").
codes_key, items_key, factors_key, scoring_key	Names of the optional metadata arrays (codes, item text, factor labels, +1/-1 scoring). Missing keys are silently skipped.

Details

The archive is expected to contain a 2-D embeddings array; the other fields are optional and matched by name.

Value

An object of class "sfa_embeddings": a list with embeddings (numeric matrix, n_items x dim, with item codes as rownames when available) and any of codes, items, factors, scoring found in the archive.

See Also

[sfa](#), [sfa_similarity](#), [sfa_corplot](#)

Examples

```
## Not run:
emb <- sfa_load_npz("DASS_items_8B.npz")
emb                                # summary of what was loaded
sfa_corplot(sfa_similarity(emb))  # grouped heatmap, two lines total
fit <- sfa(emb)                  # or run the full analysis

## End(Not run)
```

sfa_load_region	<i>Load a Saved Construct Region</i>
-----------------	--------------------------------------

Description

Reads a region saved by [sfa_build_region\(\)](#) (via its file argument or [saveRDS\(\)](#)) and validates its class.

Usage

```
sfa_load_region(file)
```

Arguments

file	Path to the saved .rds region file.
------	-------------------------------------

Value

The "sfa_region" object.

sfa_map	<i>Velicer's Minimum Average Partial for Similarity Matrices</i>
---------	--

Description

Applies Velicer's (1976) minimum average partial (MAP) test to an embedding similarity matrix. MAP extracts principal components one at a time and tracks the average squared partial correlation among the residuals; the component count at the minimum is the suggested dimensionality.

Usage

```
sfa_map(sim_matrix, max_factors = NULL)
```

Arguments

sim_matrix	Numeric similarity matrix (n_items x n_items), or a fitted "sfa" object.
max_factors	Largest component count to evaluate (default: number of items minus two). The scan also stops early if a residual variance becomes non-positive.

Details

MAP needs no sample size and no null model, so it transfers to similarity matrices without adaptation. Interpret it with care in this setting: on embedding similarity matrices MAP tends to track all reliably estimated structure, including minor components well beyond the interpretable factor count, which is why it is available in `sfa_nfactors()` but not part of the default method set.

Three guardrails can make the result diverge from Velicer's rule in edge cases, always toward retaining at least one factor: the count is floored at one (Velicer's comparison against the zero-component baseline can recommend retaining none; the baseline is reported as `map0` so that comparison remains available), the default scan stops at two below the item count (Velicer evaluates to one below), and the scan ends early when a residual variance turns non-positive (candidate counts whose partial correlations are undefined on a non-positive-semi-definite input are skipped as missing), which can truncate or thin the search before the global minimum.

Value

A list of class "sfa_map" with components:

n_factors Integer: component count at the minimum average squared partial correlation (floored at one).

map Numeric vector: the MAP criterion at each evaluated count.

map0 Baseline average squared off-diagonal correlation with no components removed.

References

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. *Psychometrika*, 41(3), 321–327. doi:[10.1007/BF02293557](https://doi.org/10.1007/BF02293557)

Examples

```
data(big5)
sim <- sfa_similarity(big5$embeddings, "mean_centered_pearson")
sfa_map(sim)
```

sfa_name

Name the Factors of an sfa Fit

Description

Retrieves a verbal label for every factor of an `sfa()` fit by embedding the fit's items under a construct-retrieval instruction, building one naming target per factor, and ranking a large pre-filtered candidate pool. Deterministic: the same fit, model, and pool always produce the same labels.

Usage

```
sfa_name(
  fit,
  model = NULL,
  pool = NULL,
  n_candidates = 5L,
  instruction = NULL,
  collision = TRUE,
  loo_sets = TRUE,
  salient = NULL,
  block_size = 50000L,
  ...
)
```

Arguments

<code>fit</code>	An object of class <code>sfa</code> .
<code>model</code>	Embedding model used for naming. Default <code>NULL</code> reuses the model the fit was embedded with. Passing a different model (e.g. a larger one) switches naming to that model while keeping the fitted factor structure - extraction and naming reward different model properties, and a larger namer typically improves label abstraction.
<code>pool</code>	An <code>sfa_pool</code> object, or <code>NULL</code> to fetch/build the pool for <code>model</code> via sfa_pool() .
<code>n_candidates</code>	Length of the gated candidate list per factor (default 5).
<code>instruction</code>	Override the naming instruction (see sfa_naming_instruction()); a warning notes that results were validated under the default.
<code>collision</code>	Resolve duplicate labels across factors via the geometric keeper (default <code>TRUE</code>).
<code>loo_sets</code>	Compute leave-one-out candidate sets (default <code>TRUE</code>).
<code>salient</code>	Loading threshold defining a factor's items for naming: items whose primary factor this is. Reserved for future use; the assignment rule is primary-loading, as in the research method.
<code>block_size</code>	Pool rows per block during retrieval (memory knob).
<code>...</code>	Passed to the embedding backend.

Value

An object of class `sfa_labels`: a `data.frame` with one row per factor (factor, label, rule, candidates, n_items, collision_moved) plus attributes `gated`, `model`, `instruction`.

sfa_nameability *Nameability of an Orientation*

Description

Runs the full `sfa_name()` selection rule (family-gated candidate walk, construct-noun preference, leave-one-out sets, geometric collision keeper) on an arbitrary pattern matrix of the lexical map's items, and returns the retrieved labels together with the nameability criterion: per factor, the cosine between the factor's naming target and its retrieved term's embedding, and their mean.

Usage

```
sfa_nameability(
  lexmap,
  loadings,
  n_candidates = 5L,
  collision = TRUE,
  loo_sets = TRUE,
  baseline = NULL,
  baseline_sd = NULL
)
```

Arguments

<code>lexmap</code>	An <code>sfa_lexmap()</code> object.
<code>loadings</code>	A pattern matrix (items x factors) in any orientation of the fitted solution.
<code>n_candidates</code>	Visible label window (default 5, as in <code>sfa_name()</code>).
<code>collision</code>	Resolve duplicate labels via the geometric keeper.
<code>loo_sets</code>	Compute leave-one-out candidate sets.
<code>baseline</code>	Optional numeric vector, one value per pool word, giving that word's expected target cosine under a null. When supplied, word scores are centered on it ("surprise scoring"), which suppresses words that sit close to every target rather than to this factor in particular. Default NULL leaves scores uncentered.
<code>baseline_sd</code>	Optional numeric vector, one value per pool word, giving the standard deviation of that word's null target cosine. When supplied together with <code>baseline</code> , scores are standardized rather than merely centered. Default NULL.

Value

A list with labels (data frame: factor, label, rule, n_items, collision_moved, row), candidates (list column), criterion (mean), and per_factor.

sfa_naming_instruction	<i>The Default Naming Instruction</i>
------------------------	---------------------------------------

Description

Returns the instruction string used to embed items for factor naming.

Usage

```
sfa_naming_instruction()
```

Value

A character scalar.

sfa_nfactors	<i>Unified Factor Retention Diagnostics</i>
--------------	---

Description

Runs multiple factor retention methods on an embedding similarity matrix and tabulates the results, mirroring the workflow of [N_FACTORS](#).

Usage

```
sfa_nfactors(  
  sim_matrix,  
  embeddings = NULL,  
  methods = "parallel",  
  seed = 42L,  
  parallel_iter = 100L,  
  max_factors = NULL,  
  rotate = "oblimin",  
  fm = "minres",  
  ...  
)
```

Arguments

- sim_matrix Numeric similarity matrix (n_items x n_items).
- embeddings Numeric embedding matrix (n_items x embedding_dim). Required when "parallel" or "EKC" is in methods.

methods	Character vector of retention methods to run. Supported: "parallel" (<code>sfa_parallel()</code>), "kaiser" (latent-root criterion), "TEFI", "EGA" (requires EGAnet), "EKC" (<code>sfa_ekc()</code>), "MAP" (<code>sfa_map()</code>), and "semk" (<code>sfa_semk()</code>), the calibrated learned rule; requires Python and a one-time model download). The default runs parallel analysis alone, matching the field's conventional retention default; request the multi-criterion battery explicitly (the package's own demonstration uses <code>c("parallel", "kaiser", "TEFI", "EGA", "EKC")</code>). Notes for choosing: EGA needs the suggested EGAnet package; the latent-root rule is retained for reference despite its known liberal bias; TEFI tends to run low on embedding similarity matrices; MAP tends to track reliable minor structure well past the interpretable factor count, which would pull the modal consensus deep; and sem-k is the only criterion with validated planted-truth error rates on embedding matrices (its classical battery votes are consumed internally as features, so a consensus mixing sem-k with those same criteria double-counts them).
seed	Random seed for parallel analysis.
parallel_iter	Iterations for parallel analysis.
max_factors	Maximum factors to test for TEFI (default: auto).
rotate	Rotation for TEFI extraction (default "oblimin").
fm	Extraction method for TEFI (default "minres").
...	Additional arguments (currently unused).

Value

An object of class "sfa_nfactors" with:

methods Data frame with one row per method: method name, suggested `n_factors`.

consensus Integer: modal recommendation across methods. When two or more recommendations tie for the mode, the smallest tied value is returned (the more parsimonious solution). With a single method this equals that method's suggestion, and `print()` omits the consensus line.

eigenvalues Numeric vector: observed eigenvalues.

parallel Parallel analysis result (if run), or NULL.

sfa_nli_matrix

Signed Item Similarity from Natural Language Inference

Description

Builds an item-by-item similarity matrix from natural language inference (NLI) rather than cosine similarity. For each ordered item pair the NLI model returns probabilities of *entailment* (E) and *contradiction* (C); the signed relation is $E - C$ (near +1 = same meaning/direction, near -1 = opposite). Unlike plain embeddings — which place antonyms close because they share a topic — NLI separates "means the same" from "means the opposite", so reverse-keyed items are handled directly (Bowman et al., 2015; Hommel & Arslan, 2025).

Usage

```
sfa_nli_matrix(
  items,
  model = "cross-encoder/nli-deberta-v3-base",
  classifier = NULL,
  symmetric = TRUE
)
```

Arguments

<code>items</code>	Character vector of item texts.
<code>model</code>	NLI cross-encoder model name (default "cross-encoder/nli-deberta-v3-base"), used by the default classifier.
<code>classifier</code>	Optional function taking two equal-length character vectors (premises, hypotheses) and returning a matrix/data frame with numeric columns entailment and contradiction (one row per pair). These are typically probabilities, but any finite numeric scores are accepted — only the signed difference entailment - contradiction is used, so the values need not lie in $[0, 1]$. Supply this to use a custom NLI backend (or for testing); the default uses a sentence-transformers CrossEncoder via reticulate .
<code>symmetric</code>	Logical: average the two directions (i,j) and (j,i) (default TRUE).

Details

The resulting matrix can be passed straight to `sfa` via its `similarity` argument.

The negative-sign-for-contradiction convention adapts Hommel and Arslan (2025), who fine-tuned a sentence-embedding model on SNLI pairs relabeled with signed similarity magnitudes. This function applies the sign convention directly at inference — entailment minus contradiction from an off-the-shelf NLI classifier — with no fine-tuning.

Value

A symmetric numeric matrix (`n_items` x `n_items`) of signed relations with 1 on the diagonal and item text as dimnames. With a probability classifier (the default) the off-diagonal values lie in $[-1, 1]$ (1 = same direction, -1 = opposite). A custom classifier returning raw (non-probability) scores may yield values outside $[-1, 1]$; these are passed through unchanged, so such a matrix may not be correlation-like and may need rescaling before `sfa(similarity = ...)`.

References

- Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 632–642). Association for Computational Linguistics. doi:10.18653/v1/D151075
- Hommel, B. E., & Arslan, R. C. (2025). Language models accurately infer correlations between psychological items and scales from text alone. *Advances in Methods and Practices in Psychological Science*, 8(4). doi:10.1177/25152459251377093

See Also[sfa](#), [sfa_similarity](#)**Examples**

```
data(big5)
# custom classifier (no Python needed) returning entailment/contradiction probs
clf <- function(premise, hypothesis) {
  same <- substr(premise, 1, 3) == substr(hypothesis, 1, 3)
  data.frame(entailment = ifelse(same, 0.8, 0.1),
             contradiction = ifelse(same, 0.05, 0.5))
}
M <- sfa_nli_matrix(big5$items[1:6], classifier = clf)
round(M, 2)

## Not run:
# default backend uses a Python NLI cross-encoder via reticulate:
M <- sfa_nli_matrix(big5$items)
fit <- sfa(big5$items, similarity = M)

## End(Not run)
```

sfa_parallel

*Embedding-Adapted Parallel Analysis***Description**

Determines the number of factors to retain from an embedding similarity matrix using random unit vectors as the null distribution, avoiding the need for a participant-level sample size.

Usage

```
sfa_parallel(
  sim_matrix,
  embeddings,
  n_iter = 100L,
  percentile = 95,
  seed = 42L
)
```

Arguments

<code>sim_matrix</code>	Numeric similarity matrix (n_items x n_items).
<code>embeddings</code>	Numeric embedding matrix (n_items x embedding_dim).
<code>n_iter</code>	Number of random iterations (default 100).
<code>percentile</code>	Percentile of null eigenvalues to use as threshold (default 95).
<code>seed</code>	Random seed, used via <code>withr::with_seed()</code> without touching the global RNG state.

Details

The adaptation keeps Horn’s (1965) logic — retain leading eigenvalues that exceed those of structureless data of the same size — but replaces the respondent-level null with similarity matrices of random Gaussian unit vectors in the item count and embedding dimension of the data. Retention follows Horn’s sequential rule: leading eigenvalues are counted until the first falls at or below its null percentile (against the 95th null percentile by default, a common modern choice; Horn compared against the null mean). Two caveats follow from the null. First, random unit vectors in a high-dimensional space are nearly orthogonal, so the null similarity matrix is near-identity and the eigenvalue thresholds concentrate just above one. Second, the null carries none of the general positive similarity component that real item embeddings share, so it is a structureless baseline, not a matched one. Benchmarking on embedding similarity matrices, Garrido et al. (2025) found conventional parallel analysis systematically overextracted; corroborate retention with the other criteria in `sfa_nfactors()`.

Value

- A list with components:
- n_factors** Integer: suggested number of factors.
- observed** Numeric vector: observed eigenvalues (descending).
- percentiles** Numeric vector: threshold eigenvalues from the null.

References

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30(2), 179–185.

Garrido, L. E., Russell-Lasalandra, L. L., & Golino, H. (2025). Estimating dimensional structure in generative psychometrics: Comparing PCA and network methods using large language model item embeddings. PsyArXiv preprint. doi:10.31234/osf.io/2s7pw_v1

sfa_pool	Fetch or Build the Candidate Pool for a Naming Model
----------	--

Description

Returns the candidate pool used by `sfa_name()`: the pre-filtered word list (369,703 label-eligible terms with precomputed word-family and dictionary-membership columns) together with its embedding matrix under the given model. Pre-generated pools are downloaded once and cached; for models without a pre-generated pool the word list is embedded locally (slow without a GPU) and cached thereafter.

Usage

```
sfa_pool(  
  model,  
  precision = c("int8", "fp16"),  
  download = interactive(),  
  build = interactive(),  
  dir = NULL  
)
```

Arguments

model	Embedding model id (as used by the sbert backend).
precision	"int8" (default) or "fp16". int8 pools are half the download and reproduce the fp16 labels on all but 3 of 75 benchmark factors (each a weak factor relabeled with a near-synonym; the diff is listed in the package documentation). Use "fp16" for exact parity with the published research pipeline. Locally built pools are always fp16.
download	Permission to download missing artifacts. Defaults to interactive(); in non-interactive sessions pass TRUE explicitly (CRAN policy: no silent large downloads).
build	Permission to embed the word list locally when no pre-generated pool exists for model. Defaults to interactive().
dir	Cache directory override (mainly for tests).

Value

An object of class sfa_pool: a list with words (data.frame: word, family, tier1), emb (matrix-like, one row per word, memory-mapped when read from disk), dim, model, precision.

sfa_project	<i>Semantic Projection onto Bipolar Axes</i>
-------------	--

Description

Places each item on a continuous scale defined by two opposing text poles (Grand et al. 2022). An axis is built as the direction from a "low" pole to a "high" pole (e.g. *mild* -> *severe*, *passive* -> *active*); every item is then projected onto that line. Unlike factor grouping (which says *which* construct an item belongs to), projection says *where along a named dimension* the item falls — useful for checking that a scale’s items span a full range of intensity/severity, ordering items, or locating items on an interpretable axis.

Usage

```
sfa_project(
  x,
  axes,
  normalize = TRUE,
  pole_embeddings = NULL,
  embed = NULL,
  model = NULL
)
```

Arguments

<code>x</code>	An "sfa" object, or a numeric item-embedding matrix (n_items x dim) with item rownames.
<code>axes</code>	A named list of axes. Each element defines the two poles, as either a named character vector <code>c(low = "...", high = "...")</code> or a list <code>list(low = c(...phrases...), high = c(...phrases...))</code> (multiple phrases per pole are averaged, which is more robust).
<code>normalize</code>	Logical. If TRUE (default), rescale each item's projection so 0 = the low pole and 1 = the high pole (values may fall outside 0 to 1). The 0-to-1 pole convention is this package's convenience, not from Grand et al., whose projections are unbounded inner products. If FALSE, return the raw cosine projection in the range -1 to 1.
<code>pole_embeddings</code>	Optional named list (one entry per axis) of precomputed pole embeddings, each a list with low and high numeric matrices/vectors. Use when <code>x</code> carries no embedding backend.
<code>embed, model</code>	Embedding backend/model for the pole text. Default to the backend/model recorded on <code>x</code> .

Details

This uses the **cosine** of each item against the pole-difference axis (a length-normalized variant of Grand et al.'s raw inner-product projection), so scores are comparable across items of differing embedding norm. As in Grand et al., a *bipolar* (two-pole) axis is what gives a diagnostic direction; a single pole is far less informative.

Value

An object of class "sfa_projection": a list with the item-by-axis scores matrix, the axis definitions, and `normalize`.

References

Grand, G., Blank, I. A., Pereira, F., & Fedorenko, E. (2022). Semantic projection recovers rich human knowledge of multiple object features from word embeddings. *Nature Human Behaviour*, 6(7), 975–987. doi:10.1038/s41562022013168

See Also

[sfa_anchor](#)

Examples

```
data(big5)
fit <- sfa(
  data.frame(code = big5$codes, item = big5$items,
    factor = big5$factors, scoring = big5$scoring),
  embeddings = big5$embeddings, scoring = big5$scoring, nfactors = 5)

# project items onto a neuroticism -> extraversion axis using precomputed poles
poles <- list(NtoE = list(
  low = big5$embeddings[big5$factors == "Neuroticism", ],
  high = big5$embeddings[big5$factors == "Extraversion", ]))
pr <- sfa_project(fit, axes = list(NtoE = c(low = "neurotic", high = "extraverted")),
  pole_embeddings = poles)
head(round(pr$scores, 2))

## Not run:
# with a live embedding backend, name the poles in words and they are embedded:
sfa_project(fit, axes = list(severity = c(low = "mild", high = "severe")))

## End(Not run)
```

sfa_redundancy	<i>Detect Redundant (Near-Duplicate) Items</i>
----------------	--

Description

Finds pairs of items that are so semantically similar they are effectively duplicates — they add length without adding information. This is distinct from [sfa_simplify](#), which removes *weak* items (far from their construct); redundancy targets *near-twin* items (very close to *each other*).

Usage

```
sfa_redundancy(x, threshold = NULL, method = c("wto", "cosine"))
```

Arguments

x	An "sfa" object (uses its similarity matrix) or a symmetric numeric item-by-item similarity matrix.
threshold	Redundancy cutoff. Item pairs with overlap at or above this value are flagged. Defaults to 0.25 for "wto" (the Unique Variable Analysis cut-off) and 0.80 for "cosine".
method	Overlap measure:

"wto" (default) Unique Variable Analysis (Christensen et al. 2023): absolute weighted topological overlap on an **EBICglasso** network, the paper's estimator. Requires the **EGAnet** package. Because an embedding similarity matrix has no response sample, the network is estimated with a nominal sample size large enough to keep the EBIC model selection in its stable regime (EBIC over-shrinks to an empty graph when the sample size equals the item count). Estimating a sparse network first is what gives wTO its discriminating power; computing it on the dense matrix compresses every pair into a narrow band.

"cosine" Direct pairwise similarity. Dependency-free and well spread for dense embedding matrices.

Value

An object of class "sfa_redundancy": a list with the flagged pairs (data frame: item_i, item_j, overlap), redundant clusters (connected groups of mutually redundant items), and suggest_remove (all-but-one item per cluster — keep one representative). Unique Variable Analysis is a *detection* method: Christensen et al. (2023) leave the handling of flagged redundancies to the researcher, so the keep-the-most-central-item suggestion (highest mean absolute similarity) is this package's convenience rule, not part of UVA.

References

Christensen, A. P., Garrido, L. E., & Golino, H. (2023). Unique Variable Analysis: A network psychometrics method to detect local dependence. *Multivariate Behavioral Research*, 58(6), 1165–1182. doi:[10.1080/00273171.2023.2194606](https://doi.org/10.1080/00273171.2023.2194606)

See Also

[sfa_simplify](#)

Examples

```
data(big5)
fit <- sfa(
  data.frame(code = big5$codes, item = big5$items,
             factor = big5$factors, scoring = big5$scoring),
  embeddings = big5$embeddings, scoring = big5$scoring, nfactors = 5)

# flag near-duplicate item pairs
sfa_redundancy(fit, threshold = 0.8, method = "cosine")
```

Description

Takes a region's sentences (from `sfa_build_region()` or `sfa_build_regions()`, including sentence-only regions built with `embeddings = FALSE`) and embeds them under the given encoder, returning a region in that encoder's space. This is how an encoder-ladder study reuses one extraction: the regions differ only in embedding space, never in text.

Usage

```
sfa_reembed_region(
  region,
  embed = "sbert",
  model = NULL,
  cache = TRUE,
  file = NULL
)
```

Arguments

`region` An "sfa_region" (or path to one).
`embed, model, cache` As in `sfa_build_region()`.
`file` Optional path to save the re-embedded region.

Value

The "sfa_region" with new embeddings and encoder.

sfa_region_bank	<i>Use a Region's Own Stored Embeddings as an Embedder</i>
-----------------	--

Description

Turns an "sfa_region" into a lookup function over its own sentences, for use as `embed =` when the "items" of an audit are drawn from a region itself. The canonical use is the cross-audit region-overlap baseline: sentences sampled from construct region A are audited as pretend items against region B, so their embeddings come from A's own stored matrix and no encoder loads. Requires the two regions to share the same instruction (regions built together always do).

Usage

```
sfa_region_bank(region)
```

Arguments

`region` An "sfa_region" (or path to one) with embeddings.

Value

A function (texts) -> matrix, for use as embed = .

sfa_semk	<i>Calibrated Semantic Factor Retention (sem-k)</i>
----------	---

Description

Estimates the number of semantic factors in an item set from its embeddings using sem-k: a learned retention rule trained on a planted-truth corpus of LLM-written item sets with known structure, embedded in realistic encoder geometry. Unlike null-referenced eigenvalue rules, sem-k is a calibrated estimator: its error rates are measured on held-out planted configurations (65.7% exact, 73.9% within 25% on the v1–v4 corpus under Qwen3-Embedding-8B), and every verdict carries a 90% split-conformal interval.

Usage

```
sfa_semk(  
  sim_matrix = NULL,  
  embeddings = NULL,  
  floor = NULL,  
  seed = 42L,  
  download = interactive(),  
  quiet = FALSE  
)
```

Arguments

sim_matrix	A fitted "sfa" object, or a similarity matrix (accepted for signature symmetry with the other retention criteria; sem-k computes its own similarity internally from the embeddings).
embeddings	Numeric embedding matrix (n_items x embedding_dim). Required unless sim_matrix is a fitted "sfa" object that carries embeddings.
floor	Register-floor calibration for the encoder that produced the embeddings (mean off-diagonal similarity of construct-dead survey-register items). NULL (default) uses the training encoder's floor (Qwen3-Embedding-8B, 0.478).
seed	Random seed for the feature-extraction bootstrap (verdicts are seed-invariant on 41 of 42 benchmark scales, max spread 1).
download	Permission to download the model artifact if not yet cached. Defaults to interactive() (CRAN policy: no silent downloads).
quiet	Suppress download progress messages.

Details

The estimand is *semantic* dimensionality: the number of distinguishable meaning clusters the items' embedding geometry supports. Across 35 scales with large response archives, semantic verdicts track empirical human-data dimensionality far better than documented textbook counts do; where the two diverge (for example, single-construct symptom inventories carrying real symptom-cluster structure), human response data typically diverges the same way. Treat sem-k as one voice alongside the granularity evidence in `sfa_dimselect()` when the interval is wide.

The rule generalizes across encoders: retrained and evaluated on nine encoders from six providers (Qwen 0.6B–8B, e5-mistral, NVIDIA llama-embed-nemotron, Microsoft harrier, OpenAI text-embedding-3 small/large, Google gemini-embedding-2), exact accuracy stays within 61.6–69.0% and real-scale verdicts agree across providers at mean pairwise Spearman .89. The shipped artifact is the Qwen3-Embedding-8B model; for other encoders pass the encoder's register floor via `floor` (see the calibration files distributed with the sem-k release).

Requires Python with numpy, scipy, scikit-learn (pinned to the 1.8 series, matching the artifact's training version), and joblib (all declared automatically via `reticulate::py_require()` on first use), and a one-time ~17 MB artifact download (cached under `tools::R_user_dir("semanticfa", "cache")`).

Value

A list of class "sfa_semK" with components:

n_factors Integer: the sem-k point estimate of semantic k.

lo90, hi90 Integer bounds of the 90% split-conformal interval (calibrated coverage 91–96% across encoders).

floor The register floor used.

battery Named integer vector: the classical battery votes (kaiser, pa_iso, ekc, map) consumed as features, for reference.

artifact Artifact identifier and training-corpus tag.

References

Yanitski, D., & Westbury, C. (in preparation). How many factors does a questionnaire mean? Validated factor retention for language-model embedding similarity matrices.

Goretzko, D., & Buhner, M. (2020). One model to rule them all? Using machine learning algorithms to determine the number of factors in exploratory factor analysis. *Psychological Methods*, 25(6), 776–786. doi:10.1037/met0000262

Examples

```
## Not run:
data(big5)
sim <- sfa_similarity(big5$embeddings, "mean_centered_pearson")
sfa_semK(sim, big5$embeddings) # 5 [2, 13]

## End(Not run)
```

sfa_similarity

Compute Embedding Similarity Matrix

Description

Transforms item embeddings into an item-by-item similarity matrix using one of several published methods.

Usage

```
sfa_similarity(
  embeddings,
  encoding = "atomic",
  scoring = NULL,
  factors = NULL,
  codes = NULL
)
```

Arguments

embeddings	Numeric matrix (n_items x embedding_dim).
encoding	Character string specifying the similarity transform: "atomic" (default), "atomic_reversed", "squid", or "mean_centered_pearson". See Details.
scoring	Numeric vector of +1/-1 per item (keying direction). Applies only to the atomic encodings (Guenole et al.); "squid" and "mean_centered_pearson" are keying-free by design, and passing scoring with real reverse-keyed (-1) items to them is ignored with a warning. If NULL, defaults to all +1 (with a message for "atomic_reversed").
factors	Optional character/factor vector of per-item subscale labels. When supplied it is <i>recorded</i> on the returned matrix (as a "factors" attribute) so that sfa_corplot can group the items; it does not reorder the matrix (rows stay aligned with the input items).
codes	Optional character vector of short item codes (e.g. "D3", "A2"). Recorded on the returned matrix (as a "codes" attribute) and used as axis labels by sfa_corplot .

Details

"atomic" (default) Cosine similarity of the item embeddings (computed by L2-normalizing internally). Equivalent to "atomic_reversed" with all +1 scoring. Named for the atomic encoding of Guenole et al., who additionally embed items separately and average within facet; this function operates on the per-item embeddings it is given.

"atomic_reversed" Multiply each embedding by its scoring direction (+1/-1) first, then cosine similarity (Guenole et al.). Use this for scales with reverse-keyed items.

"squid" Subtract the questionnaire-mean embedding (SQuID; Pellert et al. 2026), then cosine similarity (the L2-normalization is this package's similarity step; Pellert et al. define SQuID as the mean-subtraction and measure similarity by cosine or Pearson afterwards). The centering recovers negative between-dimension correlations, so this encoding is keying-free (no scoring/sign-flip). Pellert et al. note that reverse-keyed items remain an open challenge – they state that meaningfully "reversing" a semantic embedding is conceptually unclear and needs further methodological work, not that centering resolves it.

"mean_centered_pearson" Mean-center each embedding across its dimensions, L2-normalize. Cosine similarity then equals Pearson correlation, yielding a true correlation matrix (the centered-cosine = Pearson identity is attributed by Pokropek (2026) to Chen et al. (2020); see also Kmetty et al. 2021 and Casella et al. 2024). Keying-free.

Value

A symmetric numeric matrix (n_items x n_items) with 1s on the diagonal.

References

- Milano, N., Luongo, M., Ponticorvo, M., & Marocco, D. (2025). Semantic analysis of test items through large language model embeddings predicts a-priori factorial structure of personality tests. *Current Research in Behavioral Sciences*, 8, 100168. doi:10.1016/j.crbeha.2025.100168
- Casella, M., Luongo, M., Marocco, D., Milano, N., & Ponticorvo, M. (2024). LLM embeddings on test items predict post hoc loadings in personality tests. *Ital-IA 2024: 4th National Conference on Artificial Intelligence*, CEUR Workshop Proceedings.
- Guenole, N., D'Urso, E. D., Samo, A., Sun, T., & Haslbeck, J. M. B. (Preprint). Enhancing Scale Development: Pseudo Factor Analysis of Language Embedding Similarity Matrices. OSF. <https://osf.io/3mpzb/>
- Pellert, M., Lechner, C. M., Sen, I., & Strohmaier, M. (2026). Neural network embeddings recover value dimensions from psychometric survey items on par with human data (Survey and Questionnaire Item Embeddings Differentials, SQuID). *Findings of the Association for Computational Linguistics: EACL 2026*, 5738–5752.
- Pokropek, A. (2026). From keyword-based text measures to latent variables: Confirmatory factor analysis with word embeddings. *EPJ Data Science*. doi:10.1140/epjds/s13688026006541
- Chen, X., Ding, N., Levinboim, T., & Soricut, R. (2020). Improving text generation evaluation with batch centering and tempered word mover distance. *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems (Eval4NLP)*, 51–59.
- Kmetty, Z., Koltai, J., & Rudas, T. (2021). The presence of occupational structure in online texts based on word embedding NLP models. *EPJ Data Science*, 10, 55. doi:10.1140/epjds/s13688021-003119

Description

Selects a reduced (short-form) item set per group using only the items' semantic structure — no human response data — and reports how well the reduced set preserves the factor structure of the full scale (in the spirit of Wang et al., 2026; Jung & Seo, 2025). It selects items by centroid/medoid proximity within a grouping, rather than reimplementing those papers' specific clustering pipelines. The output is a **candidate** short form that should be validated psychometrically before use.

Usage

```
sfa_simplify(
  x,
  target_n,
  method = c("anchor", "medoid"),
  groups = c("theoretical", "fitted"),
  ...
)
```

Arguments

<code>x</code>	An object of class "sfa" with stored input embeddings (fit with this version of <code>sfa()</code>).
<code>target_n</code>	Integer number of items to keep per group. Groups with $\leq$ <code>target_n</code> items are kept in full.
<code>method</code>	"anchor" (default) or "medoid".
<code>groups</code>	How items are grouped before trimming: "theoretical" (default; the factor labels supplied to <code>sfa()</code>) or "fitted" (each item assigned to its strongest extracted factor — lets the groups emerge from the items, after Jung & Seo 2025, and needs no theoretical key).
<code>...</code>	Currently unused.

Details

Two selection strategies are offered:

"anchor" (default) Keep the items most similar to their own group's centroid (un-flipped, leave-one-out; see [sfa_anchor](#)); drop the weakest. Simple and interpretable, but can retain near-duplicate items (see [sfa_redundancy](#)).

"medoid" Within each group, greedily select items that are both representative (close to the group centroid) and non-redundant (spread apart in embedding space). Trades a little central tendency for broader coverage.

After selection the scale is re-fit on the kept items and compared with the full-scale solution: number of factors retained and structure recovery against the theoretical grouping (NMI and ARI).

Value

An object of class "sfa_simplify": a list with `keep` (kept item codes), `drop` (dropped items with reasons), the re-fit `reduced_fit`, and a fidelity report.

References

Wang, B., Zhang, Y., Hu, Y., Hou, H., Peng, K., & Ni, S. (2026). Discovering semantic latent structures in psychological scales: A response-free pathway to efficient simplification. arXiv:2602.12575 (preprint).

Jung, S.-J., & Seo, J.-W. (2025). A transformer-based embedding approach to developing short-form psychological measures. *Frontiers in Psychology*, 16, Article 1640864. doi:10.3389/fpsyg.2025.1640864

See Also

[sfa_anchor](#), [sfa_redundancy](#), [sfa_congruence](#)

Examples

```
data(big5)
fit <- sfa(
  data.frame(code = big5$codes, item = big5$items,
             factor = big5$factors, scoring = big5$scoring),
  embeddings = big5$embeddings, scoring = big5$scoring, nfactors = 5)

# keep the 5 most representative items per construct
short <- sfa_simplify(fit, target_n = 5, method = "anchor")
short$keep

# group by the fitted factors instead of the supplied key (needs no labels)
sfa_simplify(fit, target_n = 5, groups = "fitted")$keep
```

Index

- * **datasets**
 - big5, 5
- as_psych, 3, 4, 9
- big5, 4, 5
- cor.plot, 20
- fa, 8, 9, 28
- fa.sort, 9
- factor.congruence, 9
- N_FACTORS, 46
- plot, 34
- plot.sfa_coverage, 6
- plot.sfa_coverage(), 7
- plot.sfa_coverage_battery, 6
- saveRDS(), 13, 42
- semanticfa (semanticfa-package), 3
- semanticfa-package, 3
- sfa, 3, 7, 11, 21, 27, 28, 30, 41, 48, 49
- sfa(), 22, 39, 40, 43
- sfa_anchor, 3, 10, 36, 38, 53, 60, 61
- sfa_build_bank, 11
- sfa_build_bank(), 31
- sfa_build_region, 12
- sfa_build_region(), 14, 15, 21, 22, 42, 55
- sfa_build_regions, 14
- sfa_build_regions(), 55
- sfa_cd, 16
- sfa_clear_cache, 18
- sfa_combine_banks, 18
- sfa_congruence, 3, 9, 19, 61
- sfa_corplot, 20, 30, 33, 34, 41, 58
- sfa_coverage, 21
- sfa_coverage(), 6, 7, 11–15, 31, 32
- sfa_cross_matrix, 25
- sfa_cross_matrix(), 23
- sfa_deletion_gaps, 26
- sfa_dimselect, 3, 8, 27
- sfa_dimselect(), 57
- sfa_ekc, 29
- sfa_ekc(), 47
- sfa_embed, 3, 9, 30
- sfa_embed(), 12, 13, 18, 24, 31
- sfa_embedding_bank, 31
- sfa_embedding_bank(), 11, 18
- sfa_gaps, 32
- sfa_gaps(), 25
- sfa_install_python, 3, 32
- sfa_item_fit, 35
- sfa_itemplot, 33
- sfa_jinglejangle, 3, 36
- sfa_leximax, 38
- sfa_leximax(), 9
- sfa_lexmap, 40
- sfa_lexmap(), 9, 39, 45
- sfa_load_npz, 41
- sfa_load_npz(), 16
- sfa_load_region, 42
- sfa_load_region(), 13, 14
- sfa_map, 42
- sfa_map(), 47
- sfa_name, 43
- sfa_name(), 9, 38, 40, 45, 50
- sfa_nameability, 45
- sfa_nameability(), 39
- sfa_naming_instruction, 46
- sfa_naming_instruction(), 40, 44
- sfa_nfactors, 3, 9, 46
- sfa_nfactors(), 29, 43, 50
- sfa_nli_matrix, 3, 8, 32, 47
- sfa_parallel, 3, 9, 49
- sfa_parallel(), 29, 47
- sfa_pool, 50
- sfa_pool(), 9, 40, 44
- sfa_project, 3, 51

sfa_redundancy, [3](#), [36](#), [53](#), [60](#), [61](#)
sfa_reembed_region, [54](#)
sfa_reembed_region(), [15](#)
sfa_region_bank, [55](#)
sfa_region_bank(), [18](#)
sfa_semk, [56](#)
sfa_semk(), [8](#), [47](#)
sfa_similarity, [3](#), [8](#), [9](#), [20](#), [21](#), [27](#), [28](#), [41](#),
[49](#), [58](#)
sfa_similarity(), [17](#), [29](#)
sfa_simplify, [3](#), [11](#), [36](#), [53](#), [54](#), [59](#)
sfa_tsneplot(sfa_itemplot), [33](#)

with_seed, [8](#)
withr::with_seed(), [16](#), [49](#)