Package: semanticfa (via r-universe)

June 16, 2026

Title Semantic Factor Analysis of Language Model Embeddings

Version 0.1.0

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. Supports embedding-adapted parallel analysis, several similarity transforms (atomic reversed, SQuID centering, mean-centered Pearson), and fit diagnostics tailored to embedding matrices (TEFI, RMSR, CAF, McDonald's omega). 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 GPArotation, grDevices, graphics, psych, reticulate (>= 1.41.0), Rtsne, stats, utils, uwot, withr

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs libpng-dev python3

Repository https://devon7y.r-universe.dev

Date/Publication 2026-06-08 23:55:10 UTC

RemoteUrl https://github.com/devon7y/semanticfa

RemoteRef HEAD

RemoteSha df270c5e538160164277bb71a7de74900e15e850

Contents

`semanticfa-package` *semanticfa: Semantic Factor Analysis of Language Model Embeddings*

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 *IPIP Big Five 50-Item Inventory with Sentence-BERT Embeddings*

Description

A bundled example dataset containing the 50-item IPIP Big Five personality inventory with precomputed sentence-BERT embeddings. 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, ..., O10).

factors Character vector (length 50): theoretical factor labels.

scoring Numeric vector (length 50): +1 or -1 keying direction.

embeddings Numeric matrix (50 x 384): precomputed embeddings from the `all-MiniLM-L6-v2` sentence-BERT model.

Source

Items from the International Personality Item Pool (IPIP; https://ipip.ori.org/), which is in the public domain. Embeddings were generated with the `sentence-transformers/all-MiniLM-L6-v2` model (https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2), released under the Apache License 2.0. The regeneration script is in `data-raw/big5.R`.

Examples

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

`sfa` *Semantic Factor Analysis*

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,
  ...
)
```

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 n_factors_method.
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, print() 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", or "TEFI".
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.)

calibrate_iter	Iterations for calibration.
...	Additional arguments passed to fa.

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)
fit <- sfa(big5$items, embeddings = big5$embeddings, scoring = big5$scoring)
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. Items are sign-aligned by their scoring direction first (embeddings encode topic, not valence, so a reverse-keyed item would otherwise point away from its construct), so 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 unique(factors)) 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 (sign-aligned) 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. Cleanest for the default "atomic_reversed" and "atomic" encodings.

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_clear_cache *Clear Embedding Cache*

Description

Removes all cached embedding files created by sfa_embed().

Usage

```
sfa_clear_cache()
```

Value

Invisible NULL.

`sfa_congruence` *Compare Semantic and Empirical Factor Structures*

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 `psych::fa` 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"`.

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_dimselect *Embedding-Dimension Selection by EGA Depth Optimization*

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 reimplementation 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_reversed",
  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_reversed"`). 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 `c(nmi=, tefi=)` for the composite (default `c(nmi = 0.70, tefi = 0.30)`).
`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 richer item pools (more than ~15 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_embed` *Embed Item Text with a Language Model*

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_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_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

`x`	An object of class `"sfa"` carrying theoretical factor labels and stored (raw) embeddings.
`item`	Character vector of one or more candidate items to vet.
`construct`	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.
`reverse_key`	Logical; set `TRUE` if the candidate is a reverse-keyed item (its embedding is flipped before comparison). Default `FALSE`.
`redundancy_cutoff`	Similarity to the nearest existing item at or above which the candidate is flagged as a near-duplicate. Default 0.90.
`embed`, `model`	Embedding backend and model used to embed the candidate(s) and the construct names. Default to those recorded on `x`.

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:
fit <- sfa(dass_df, nfactors = 3)            # fit with a real embedding model
sfa_item_fit(fit, "I am sad all the time", construct = "Depression")
sfa_item_fit(fit, c("I feel calm and relaxed",
                    "My heart was racing"))   # vet several at once

## End(Not run)
```

---

sfa_itemplot *2-D Item Map (t-SNE, UMAP, PCA, or MDS)*

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, 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_jinglejangle *Detect Jingle and Jangle Fallacies Across Scales*

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).
`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_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_nfactors` *Unified Factor Retention Diagnostics*

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 = c("parallel", "kaiser", "TEFI"),
  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"` is in `methods`.
`methods`	Character vector of retention methods to run. Supported: `"parallel"`, `"kaiser"`, `"TEFI"`, `"EGA"`.
`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.

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

items	Character vector of item texts.
model	NLI cross-encoder model name (default "cross-encoder/nli-deberta-v3-base"), used by the default classifier.
classifier	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**.
symmetric	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.

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

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

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.

Yanitski, D. & Westbury, C. (2025). Embedding-adapted parallel analysis for semantic factor analysis.

---

sfa_project *Semantic Projection onto Bipolar Axes*

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

x	An "sfa" object, or a numeric item-embedding matrix (n_items x dim) with item rownames.
axes	A named list of axes. Each element defines the two poles, as either a named character vector c(low = "...", high = "...") or a list list(low = c(...phrases...), high = c(...phrases...)) (multiple phrases per pole are averaged, which is more robust).
normalize	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). If FALSE, return the raw cosine projection in the range -1 to 1.
pole_embeddings	Optional named list (one entry per axis) of precomputed pole embeddings, each a list with low and high numeric matrices/vectors. Use when x carries no embedding backend.
embed, model	Embedding backend/model for the pole text. Default to the backend/model recorded on x.

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` *Detect Redundant (Near-Duplicate) Items*

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).

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

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")
```

---

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 *recorded* 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) L2-normalize each embedding, then cosine similarity. Equivalent to `"atomic_reversed"` with all +1 scoring.

`"atomic_reversed"` Multiply each embedding by its scoring direction (+1/-1) first, L2-normalize, 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), L2-normalize, then cosine similarity. 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

`sfa_simplify` *Response-Free Scale Simplification*

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

`x`	An object of class `"sfa"` with stored input embeddings (fit with this version of `sfa()`).
`target_n`	Integer number of items to keep per group. Groups with `<= target_n` items are kept in full.

method	"anchor" (default) or "medoid".
groups	How items are grouped before trimming: "theoretical" (default; the factor labels supplied to sfa()) 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).
...	Currently unused.

Details

Two selection strategies are offered:

"anchor" (default) Keep the items most similar to their own group's centroid (sign-aligned, 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