

Package: SOMevidence (via r-universe)

October 11, 2026

Title Evidence Workflows for Self-Organizing Map Classification

Version 1.2.0

Description Provides explicit data-design objects, leakage-safe preprocessing, design-aware resampling, reproducible ensembles of self-organizing maps, and evidence-oriented diagnostics for ecological, environmental and evolutionary applications. The package separates continuous-representation diagnostics from the defensibility of a hard partition and supports explicit abstention when evidence is insufficient. Self-organizing maps follow Kohonen (1982) <[doi:10.1007/BF00337288](https://doi.org/10.1007/BF00337288)>, using the 'kohonen' implementation described by Wehrens and Kruisselbrink (2018) <[doi:10.18637/jss.v087.i07](https://doi.org/10.18637/jss.v087.i07)>.

License GPL (>= 3)

URL <https://shaowen-ye.github.io/SOMevidence/>,
<https://shaowen-ye.r-universe.dev/SOMevidence>,
<https://github.com/shaowen-ye/SOMevidence>

BugReports <https://github.com/shaowen-ye/SOMevidence/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.2.0)

Imports clue, kohonen, stats, withr (>= 2.5.0)

Suggests future, future.apply, ggplot2, knitr, mclust, palmerpenguins, rmarkdown, shiny, testthat (>= 3.0.0), yaml

Config/testthat/edition 3

Config/Needs/coverage covr

Config/Needs/documentation roxygen2

Config/Needs/lint lintr

Config/Needs/validation jsonlite, pkgload, renv

Config/Needs/website pkgdown

VignetteBuilder knitr, rmarkdown

Config/roxygen2/version 8.1.0

Repository <https://shaowen-ye.r-universe.dev>

Date/Publication 2026-08-11 14:11:37 UTC

RemoteUrl <https://github.com/shaowen-ye/SOMevidence>

RemoteRef v1.2.0

RemoteSha dfc9330d33689c020bc6e195b06d7d8632389c23

Contents

assess_defensibility	3
audit_som	4
audit_som_representation	4
audit_transfer	6
compare_cross_models	6
consensus_som	7
evaluate_external_labels	8
expand_som_spec	9
fit_cross_models	10
fit_som_ensemble	11
launch_som_app	12
map_som_ensemble	13
open_data_registry	13
pareto_candidates	14
partition_som	15
plot.som_audit	16
plot.som_consensus	16
plot.som_cross_comparison	17
plot.som_newdata_mapping	17
plot.som_partitions	18
plot.som_sensitivity	18
plot.som_transfer_audit	19
plot_som_plane	19
run_som_sensitivity	20
run_som_workflow	21
simulate_som_scenario	23
som_data	24
som_gate	25
som_preprocess	27
som_resamples	28
som_spec	29
summary.som_workflow	30
upgrade_som_object	31

Index

32

assess_defensibility *Assess partition defensibility without forcing a verdict*

Description

In addition to analyst-defined thresholds, the assessment checks that every source partition and the consensus retain the requested k, and that all requested cross-model fits succeeded. Missing audit evidence produces an uncertain result rather than being treated as a pass.

Usage

```
assess_defensibility(
  audit,
  partitions = NULL,
  k = NULL,
  gate = NULL,
  consensus = NULL,
  cross_model = NULL
)
```

Arguments

audit	A som_audit object.
partitions	Optional som_partitions object. At least one partition- derived evidence object is required for a "supported" decision.
k	Candidate number of clusters when partition stability is assessed.
gate	Optional analyst-defined som_gate. Without one the status is "not_assessed".
consensus	Optional som_consensus object for the same k.
cross_model	Optional som_cross_comparison object.

Details

A "supported" status means only that the analyst-specified evidence requirements and structural completeness checks were met for the stated data, analysis design, preprocessing, model budget and candidate k. It does not demonstrate that discrete ecological types exist, identify a causal mechanism, or establish ecological truth.

A representation audit alone cannot support a hard partition. A supported decision also requires traceable comparative partition evidence and at least one prespecified partition-quality requirement. Missing evidence or a representation-only gate produces an "uncertain" result.

Value

A som_defensibility object containing evidence and reasons.

audit_som

Audit representation quality across a SOM ensemble

Description

Audit representation quality across a SOM ensemble

Usage

```
audit_som(ensemble)
```

Arguments

ensemble A fitted som_ensemble.

Value

A som_audit object containing fit-level metrics and summaries.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 1)
specification <- som_spec(c(3, 2), seeds = 1:2, rlen = 10, k = 2)
audit_som(fit_som_ensemble(data, specification, keep_models = FALSE))
```

audit_som_representation

Audit reproducibility of continuous SOM representations

Description

Computes scope-specific representation diagnostics and compares the topology induced by pre-specified pairs of SOM fits. Pairwise evidence uses shortest-hop distances on each SOM grid, so no node labels or map orientations are forced into correspondence. All calculations are exact under explicit fit-pair and sample-pair budgets.

Usage

```
audit_som_representation(
  ensemble,
  scope = c("analysis", "assessment"),
  neighbourhood_size = NULL,
  pairs = NULL,
  max_pairwise_comparisons = 10000L,
  max_sample_pairs = 1000000L,
  fail_fast = FALSE
)
```

Arguments

ensemble	A fitted som_ensemble object. Retained kohonen model objects are not required.
scope	Rows compared within each fit: "analysis" or "assessment".
neighbourhood_size	Optional positive integer q for exact local sample-neighbourhood Jaccard. All samples tied at the q-th shortest-hop distance are included. No scientific default is supplied.
pairs	Optional prespecified data frame with fit_a and fit_b columns. The default compares all successful fits.
max_pairwise_comparisons	Maximum number of fit pairs. The function stops before dense topology calculations when this budget is exceeded.
max_sample_pairs	Maximum total number of unordered jointly mapped sample pairs across all requested fit comparisons.
fail_fast	Whether to stop at the first metric-computation error.

Details

This experimental audit concerns continuous representation, not the defensibility of a hard partition. It supplies no score, ranking, automatic threshold or decision status and is not accepted by [assess_defensibility\(\)](#).

Value

A som_representation_audit object with fit metrics, pairwise topology reproducibility, optional sample-neighbourhood records, descriptive summaries, computation budgets, provenance, warnings and failures. Sample pairs and fit pairs are not treated as independent inferential replicates.

Examples

```
data <- simulate_som_scenario("gradient", n = 45, p = 3, seed = 21)
ensemble <- fit_som_ensemble(
  data,
  som_spec(c(3, 2), seeds = 22:23, rlen = 10, k = 2),
  keep_models = FALSE
)
audit_som_representation(ensemble, neighbourhood_size = 5)
```

audit_transfer	<i>Audit transfer to held-out sampling domains</i>
----------------	--

Description

For each successful fit with assessment rows, the function contrasts mapping error in the analysis and assessment sets and records the fraction of held- out samples mapped to units unoccupied during training. These quantities describe domain shift; they do not establish ecological transferability on their own.

Usage

```
audit_transfer(ensemble)
```

Arguments

ensemble A fitted som_ensemble containing non-empty assessment sets.

Value

A som_transfer_audit object with fit-level transfer diagnostics.

Examples

```
data <- simulate_som_scenario("gradient", n = 60, p = 3, seed = 7)
splits <- som_resamples(data, method = "leave_domain_out")
specification <- som_spec(c(3, 2), seeds = 1, rlen = 10, k = 2)
ensemble <- fit_som_ensemble(data, specification, splits)
audit_transfer(ensemble)
```

compare_cross_models	<i>Compare SOM partitions with controlled cross-model references</i>
----------------------	--

Description

Compare SOM partitions with controlled cross-model references

Usage

```
compare_cross_models(
  partitions,
  cross_models,
  scope = c("analysis", "assessment", "all")
)
```

Arguments

partitions	A som_partitions object.
cross_models	A som_cross_models object.
scope	Compare analysis rows ("analysis"), assessment rows when available ("assessment") or all mapped rows ("all"). The analysis scope is the default for hard-partition defensibility; assessment scope is a separate mapping-transfer diagnostic.

Value

A som_cross_comparison containing ARI and AMI effect sizes, summaries, reference-fit success rates, source-partition completeness and source-partition provenance. Neither agreement metric is reported as accuracy.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 5)
specification <- som_spec(c(3, 2), seeds = 1:2, rlen = 10, k = 2)
ensemble <- fit_som_ensemble(data, specification, keep_models = FALSE)
partitions <- partition_som(ensemble)
references <- fit_cross_models(ensemble, methods = "ward")
compare_cross_models(partitions, references)
```

consensus_som

*Construct a sample-level consensus partition***Description**

Partitions with the same k are compared through sample co-assignment. Labels are aligned to the consensus partition only after fitting. This label alignment does not align SOM nodes and must not be used to force maps with different grids into a common node geometry.

Usage

```
consensus_som(
  partitions,
  k,
  linkage = "average",
  method = c("auto", "coassignment", "aligned_vote"),
  max_coassignment_n = 5000L
)
```

Arguments

partitions	A som_partitions object.
k	Number of clusters to audit.
linkage	Linkage used to partition the sample co-assignment matrix.

method Consensus algorithm. "auto" uses full co-assignment only when every partition assigns every sample and the sample count does not exceed max_coassignment_n. It uses medoid-aligned voting for analysis-scoped resamples with incomplete assignments or larger data.

max_coassignment_n Largest sample count for an automatically created dense co-assignment matrix. Set this deliberately because memory grows quadratically with sample count.

Value

A som_consensus object containing aligned assignments, partition-assignment, consensus-label and replicated coverage, membership support, assignment entropy and clusterwise Jaccard values. The object also records whether the consensus itself retains all requested clusters and retains the originating partition method as provenance. For co-assignment consensus, coassignment and tree contain the dense matrix and its hierarchical clustering; both are NULL for aligned voting. Aligned voting avoids quadratic growth in sample count, but alignment and upstream partition auditing can still grow quadratically in the number of ensemble members.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 3)
specification <- som_spec(c(3, 2), seeds = 1:2, rlen = 10, k = 2)
ensemble <- fit_som_ensemble(data, specification, keep_models = FALSE)
consensus_som(partition_som(ensemble), k = 2)
```

evaluate_external_labels

Compare a consensus partition with external ecological labels

Description

External labels are evaluated only after SOM fitting and consensus construction. ARI and AMI quantify agreement; neither is called accuracy, and agreement does not make the external labels ecological ground truth.

Usage

```
evaluate_external_labels(
  consensus,
  labels = NULL,
  exclude = NULL,
  label_ids = NULL,
  match_by = c("auto", "id", "position")
)
```

Arguments

consensus	A som_consensus object.
labels	Optional external labels. By default, the external_label field supplied to <code>som_data()</code> is used. Supplied labels should be matched by stable sample IDs through label_ids or complete names(labels).
exclude	Optional label values to omit, such as an explicit unknown category. Missing labels and samples without at least two analysis-scope assignments are always omitted and counted because one assignment cannot provide consensus evidence.
label_ids	Optional unique sample identifiers corresponding to labels. They may be in any order and may describe a subset of consensus samples. IDs not present in the consensus are rejected.
match_by	Matching rule. "auto" uses IDs when available. For an unnamed full-length label vector it warns and retains legacy positional matching. Use "id" to require identity matching or "position" to make full-length positional matching explicit.

Value

A som_external_assessment object containing the contingency table, long-form cluster composition, ARI, AMI, resolved matching information and mutually exclusive sample-omission accounting.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 9)
ensemble <- fit_som_ensemble(
  data, som_spec(c(3, 2), seeds = 10:11, rlen = 10, k = 2)
)
consensus <- consensus_som(partition_som(ensemble), k = 2)
labels <- stats::setNames(
  data$metadata$external_label,
  data$metadata$id
)
evaluate_external_labels(consensus, labels[sample(names(labels))])
```

expand_som_spec

Expand an ensemble specification into its model budget

Description

Expand an ensemble specification into its model budget

Usage

```
expand_som_spec(spec)
```

Arguments

spec A som_spec object.

Value

A data frame with one row per grid and random seed.

Examples

```
expand_som_spec(som_spec(c(4, 3), seeds = 1:3, rlen = 20, k = 2:3))
```

fit_cross_models	<i>Fit controlled cross-model reference partitions</i>
------------------	--

Description

K-means, Ward.D2 and Gaussian mixture models use the same data object, resampling indices and leakage-safe preprocessing as a fitted SOM ensemble. The function creates reference partitions for triangulation; it does not rank algorithms or define any method as ground truth.

Usage

```
fit_cross_models(
  ensemble,
  methods = c("kmeans", "ward"),
  k = ensemble$spec$k,
  kmeans_seeds = 1L,
  kmeans_nstart = 50L,
  kmeans_iter_max = 100L,
  gmm_model_names = NULL,
  keep_models = FALSE,
  fail_fast = FALSE,
  gmm_seed = 1L
)
```

Arguments

ensemble	A fitted som_ensemble defining data, preprocessing and resampling.
methods	Any of "kmeans", "ward" and "gmm".
k	Candidate numbers of clusters.
kmeans_seeds	Random seeds used only for K-means initialization.
kmeans_nstart	Number of random starts per K-means fit.
kmeans_iter_max	Maximum number of iterations per K-means start.

gmm_model_names	Optional covariance models passed to <code>mclust::Mclust()</code> . BIC selection occurs within each analysis split.
keep_models	Whether to retain fitted cross-model objects.
fail_fast	Whether to stop at the first fit failure.
gmm_seed	Base random seed for reproducible GMM initialization. A deterministic fit-specific seed is derived from the split and candidate k, and is retained in the result object.

Value

A `som_cross_models` object with partitions, warnings and explicit failures.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 4)
specification <- som_spec(c(3, 2), seeds = 1:2, rlen = 10, k = 2)
ensemble <- fit_som_ensemble(data, specification, keep_models = FALSE)
fit_cross_models(ensemble, methods = c("kmeans", "ward"))
```

fit_som_ensemble	<i>Fit a reproducible ensemble of self-organizing maps</i>
------------------	--

Description

Every model receives preprocessing parameters estimated only from its own analysis rows. Failures are recorded instead of silently discarded. Topographic error is computed during fitting so duplicate full processed matrices need not be retained in every ensemble member.

Usage

```
fit_som_ensemble(
  data,
  spec,
  resamples = NULL,
  preprocess = som_preprocess(),
  keep_models = TRUE,
  fail_fast = FALSE,
  parallel = FALSE
)
```

Arguments

data	A <code>som_data</code> object.
spec	A <code>som_spec</code> object.
resamples	A <code>som_resamples</code> object. The default fits the full data.

preprocess	One som_preprocess object or a named object per layer.
keep_models	Whether to retain fitted kohonen objects.
fail_fast	Whether to stop at the first model failure.
parallel	Whether to distribute ensemble members through the current future plan using future.apply. The package never changes that plan. Explicit model seeds retain reproducibility across sequential and future execution. Avoid combining worker-level parallelism with spec\$cores > 1 unless nested parallelism is intended.

Value

A som_ensemble object.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 1)
specification <- som_spec(c(3, 2), seeds = 1:2, rlen = 10, k = 2)
ensemble <- fit_som_ensemble(data, specification, keep_models = FALSE)
ensemble
```

launch_som_app

Launch the optional reproducible SOM interface

Description

The Shiny interface exposes a compact subset of the package workflow for teaching and exploratory configuration. Every completed run can export its exact R script and a YAML configuration snapshot. Version 1.1.x cannot import that snapshot to restore GUI controls. The exported script, not the GUI session or snapshot, is the executable analysis record. The interface is designed for a local R session, and SOMEvidence sends no telemetry. In a local session, selected files remain on the local computer. A remotely deployed Shiny application transfers selected files to its host, whose operator is responsible for access controls and data handling.

Usage

```
launch_som_app()
```

Value

A shiny.appobj. Pass it to `shiny::runApp()` or print it in an interactive R session. The application does not return an analysis result; users can export the executable R script for a configured run.

Lifecycle

Experimental. The interface and its exported configuration snapshot schema may change after independent usability testing. The ordinary R API remains the authoritative analysis interface.

map_som_ensemble	<i>Map new observations through a fitted SOM ensemble</i>
------------------	---

Description

Every ensemble member applies only the transformations, centering, scaling and layer geometry learned from its own analysis split. The function reports best-matching units and mapping distances separately by fit; it does not align nodes across different grids or call the result prediction accuracy.

Usage

```
map_som_ensemble(ensemble, new_data, fail_fast = FALSE)
```

Arguments

ensemble	A som_ensemble fitted with keep_models = TRUE.
new_data	A som_data object with the same named layers and variables as the training data. Metadata may differ and is retained.
fail_fast	Whether to stop at the first member-specific mapping error.

Value

A som_newdata_mapping object with sample-by-fit records, fit-level distance and occupancy summaries, failures and new-data metadata.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 8)
specification <- som_spec(c(3, 2), seeds = 1, rlen = 10, k = 2)
ensemble <- fit_som_ensemble(data, specification, keep_models = TRUE)
new_data <- som_data(layers = list(
  environment = data$layers$environment[1:3, , drop = FALSE]
))
map_som_ensemble(ensemble, new_data)
```

open_data_registry	<i>List governed tutorial and candidate open-data sources</i>
--------------------	---

Description

The registry records provenance, version, licence, implementation status and the intended role of each external dataset. Only records marked implemented are used by a current package tutorial. Candidate records are governed prospects, not claims of completed package validation. The function returns metadata only; it never downloads data or changes the user's files.

Usage

```
open_data_registry(role = NULL, domain = NULL)
```

Arguments

role	Optional role used to filter the registry. Matching is exact and case-sensitive; unmatched values return no matching rows.
domain	Optional scientific domain used to filter the registry. Matching is exact and case-sensitive; unmatched values return no matching rows.

Value

A data frame with one row per governed external dataset.

Examples

```
open_data_registry(role = "quickstart")
```

pareto_candidates	<i>Identify Pareto-efficient SOM candidates</i>
-------------------	---

Description

When `x` is a `som_audit`, fit-level metrics are directly comparable only when every successful fit uses the same analysis rows. To preserve the established interface, an audit with distinct analysis sets still returns its exploratory fit-level frontier, but emits a warning and marks the result with `attr(result, "comparison_scope") == "distinct_analysis_sets"`. Such a frontier must not be used to choose a configuration. Instead, construct prespecified configuration-level summaries over identical successful split and seed coverage, then pass that justified comparable data frame.

Usage

```
pareto_candidates(
  x,
  metrics = c(quantization_error = "min", topographic_error = "min", empty_unit_rate =
    "min")
)
```

Arguments

x	A <code>som_audit</code> object or a data frame of comparable candidates.
metrics	Named character vector whose values are "min" or "max".

Value

The non-dominated rows, with a `pareto` column. Results derived from a `som_audit` also carry a `comparison_scope` attribute.

partition_som

*Form and compare candidate hard partitions of SOM units***Description**

Form and compare candidate hard partitions of SOM units

Usage

```
partition_som(
  ensemble,
  k = ensemble$spec$k,
  method = "ward.D2",
  scope = c("analysis", "all"),
  max_pairwise_comparisons = 1000000L
)
```

Arguments

ensemble	A fitted som_ensemble.
k	Candidate numbers of clusters. Defaults to the ensemble specification.
method	Hierarchical clustering method used on SOM codebooks.
scope	Evidence scope. The default, "analysis", masks every row not used to fit a given ensemble member. "all" explicitly audits the stability of mapped labels for all rows and must not be used as training- partition evidence in assess_defensibility() .
max_pairwise_comparisons	Maximum number of pairwise partition comparisons created across all requested k. This explicit budget guards against quadratic growth in the number of ensemble members.

Value

A som_partitions object. Agreement is reported as ARI and AMI, not accuracy. AMI is chance-adjusted and normalized by the arithmetic mean of the two partition entropies.

Examples

```
data <- simulate_som_scenario("clusters", n = 45, p = 3, seed = 2)
specification <- som_spec(c(3, 2), seeds = 1:2, rlen = 10, k = 2)
ensemble <- fit_som_ensemble(data, specification, keep_models = FALSE)
partition_som(ensemble)
```

plot.som_audit	<i>Plot SOM audit trade-offs</i>
----------------	----------------------------------

Description

Plot SOM audit trade-offs

Usage

```
## S3 method for class 'som_audit'
plot(x, ...)
```

Arguments

x	A som_audit object.
...	Unused.

Value

A ggplot object.

plot.som_consensus	<i>Plot sample-level consensus evidence</i>
--------------------	---

Description

Plot sample-level consensus evidence

Usage

```
## S3 method for class 'som_consensus'
plot(
  x,
  type = c("support", "heatmap"),
  samples = NULL,
  max_samples = 500L,
  ...
)
```

Arguments

x	A som_consensus object.
type	Either "support" or "heatmap". A heatmap requires a co-assignment consensus; aligned voting does not create a dense co-assignment matrix.
samples	Optional sample indices for a heatmap.
max_samples	Maximum heatmap size allowed without explicit samples.
...	Unused.

Value

A ggplot object.

```
plot.som_cross_comparison
```

Plot cross-model agreement without ranking methods

Description

Plot cross-model agreement without ranking methods

Usage

```
## S3 method for class 'som_cross_comparison'
plot(x, ...)
```

Arguments

x	A som_cross_comparison object.
...	Unused.

Value

A faceted ggplot object showing ARI and AMI distributions.

```
plot.som_newdata_mapping
```

Plot new-data mapping shift across an ensemble

Description

Plot new-data mapping shift across an ensemble

Usage

```
## S3 method for class 'som_newdata_mapping'
plot(x, ...)
```

Arguments

x	A som_newdata_mapping object.
...	Unused.

Value

An editable ggplot object.

plot.som_partitions *Plot partition reproducibility across candidate class counts*

Description

Plot partition reproducibility across candidate class counts

Usage

```
## S3 method for class 'som_partitions'
plot(x, ...)
```

Arguments

x	A som_partitions object.
...	Unused.

Value

A faceted ggplot object with ensemble agreement intervals.

plot.som_sensitivity *Plot sample-level sensitivity across prespecified scenarios*

Description

Each point represents one sample at one candidate k. The horizontal axis conditions cluster-member sets on jointly evaluable shared samples and therefore isolates repartitioning. The vertical axis also includes eligible members unique to either scenario and therefore combines data coverage with repartitioning. Values are medians across evaluable contrasts. Membership sets are compared directly, so the plot is invariant to arbitrary cluster-label permutations. It describes sensitivity to the scenarios that were supplied, not a probability that a cluster assignment is correct.

Usage

```
## S3 method for class 'som_sensitivity'
plot(x, k = NULL, ...)
```

Arguments

x	A som_sensitivity object.
k	Optional candidate cluster counts to display.
...	Unused.

Value

An editable ggplot object.

plot.som_transfer_audit

Plot mapping shift in held-out domains

Description

Plot mapping shift in held-out domains

Usage

```
## S3 method for class 'som_transfer_audit'
plot(x, ...)
```

Arguments

x	A som_transfer_audit object.
...	Unused.

Value

A ggplot object.

plot_som_plane

Plot an interpretable plane from one SOM ensemble member

Description

This function deliberately plots a named ensemble member instead of silently treating one fitted map as representative of the ensemble. Component values are on the processed scale used for training.

Usage

```
plot_som_plane(
  x,
  type = c("component", "occupancy", "neighbour_distance"),
  fit_id = NULL,
  layer = NULL,
  variables = NULL
)
```

Arguments

x	A som_ensemble object.
type	One of "component", "occupancy" or "neighbour_distance".
fit_id	Identifier of one successful ensemble fit. It may be omitted only when the ensemble contains exactly one successful fit.
layer	Layer name used for a component plane. It may be omitted for a single-layer analysis.
variables	Optional component names. All variables in layer are shown by default.

Value

An editable ggplot object.

run_som_sensitivity	<i>Run prespecified SOM sensitivity scenarios</i>
---------------------	---

Description

Each named scenario must provide data and spec, and may provide resamples, preprocess and k. Scenarios may represent alternative variable modules, transformations, layer weights, grids or data-coverage rules. Results remain separate by scenario and evidence stream; the function creates no aggregate ranking.

Usage

```
run_som_sensitivity(
  scenarios,
  cross_models = c("kmeans", "ward"),
  cross_model_control = list(),
  consensus_method = "auto",
  max_coassignment_n = 5000L,
  max_pairwise_comparisons = 1000000L,
  sample_profiles = TRUE,
  keep_workflows = TRUE,
  fail_fast = FALSE
)
```

Arguments

scenarios	A named list of workflow argument lists.
cross_models	Reference methods used in every scenario.
cross_model_control	Named reference-model controls passed to every scenario through run_som_workflow() .
consensus_method	Consensus method used in every scenario.

max_coassignment_n	Dense co-assignment limit.
max_pairwise_comparisons	Pairwise partition-comparison budget passed to each scenario.
sample_profiles	Whether to compute sample-level membership-set contrasts. Set to FALSE when the number of scenarios and samples would make the pairwise long table unnecessarily large.
keep_workflows	Whether to retain full workflow objects.
fail_fast	Whether to stop at the first scenario failure.

Value

A `som_sensitivity` object with tidy evidence tables, shared-sample comparisons between scenario consensus partitions, sample-level membership-set comparisons, scenario failures and optionally full workflows. Scenario agreement uses only samples assigned by at least two ensemble members in both scenarios, and at least two jointly evaluable samples are required before co-membership can be compared. For each focal sample, `membership_jaccard_shared` conditions on the jointly evaluable sample universe and isolates repartitioning, whereas `membership_jaccard_all` also includes eligible members unique to either scenario and therefore combines coverage and repartitioning effects. Neither requires aligning arbitrary cluster labels. `sample_summary` reports both distributions and distinguishes coverage of all planned contrasts from coverage conditional on globally comparable contrasts. These are sensitivity diagnostics, not confidence probabilities. Comparisons across different data objects require supplied identifiers or explicit row names; unsafe default identifiers, failed scenarios, unavailable consensus and absent common `k` values remain visible through `comparison_status` rather than being silently omitted. Legacy objects without identifier provenance must be reconstructed with an explicit `id` argument before cross-object comparison. Candidate `k` values are read exactly from each scenario or its specification; malformed values are not coerced. A failed scenario without a valid candidate set remains visible with `k = NA` but cannot contribute to a candidate-specific contrast denominator.

Lifecycle

Experimental. The scenario-list interface may change after independent usability testing. Individual workflow functions and their evidence tables are the candidate-stable interfaces.

run_som_workflow	<i>Run a complete SOM defensibility workflow</i>
------------------	--

Description

The workflow preserves representation quality, partition stability, sample-level consensus and cross-model agreement as separate evidence streams. It does not combine them into a score or select a winning `k`.

Usage

```
run_som_workflow(
  data,
  spec,
  resamples = NULL,
  preprocess = som_preprocess(),
  k = spec$k,
  cross_models = c("kmeans", "ward"),
  cross_model_control = list(),
  consensus_method = "auto",
  max_coassignment_n = 5000L,
  keep_models = FALSE,
  fail_fast = FALSE,
  parallel = FALSE,
  max_pairwise_comparisons = 1000000L
)
```

Arguments

<code>data</code>	A <code>som_data</code> object.
<code>spec</code>	A <code>som_spec</code> object.
<code>resamples</code>	Optional <code>som_resamples</code> object. If <code>NULL</code> , a single full-data split is used; that run does not assess perturbation under resampling.
<code>preprocess</code>	One <code>som_preprocess</code> object or a named object per layer.
<code>k</code>	Candidate numbers of hard partitions.
<code>cross_models</code>	Controlled reference methods passed to fit_cross_models() . Use <code>character()</code> to skip cross-model fitting.
<code>cross_model_control</code>	Named list of optional <code>kmeans_seeds</code> , <code>kmeans_nstart</code> , <code>kmeans_iter_max</code> , <code>gmm_model_names</code> or <code>gmm_seed</code> settings forwarded to fit_cross_models() . Methods, candidate <code>k</code> , model retention and failure policy remain controlled by this workflow.
<code>consensus_method</code>	Consensus method passed to consensus_som() .
<code>max_coassignment_n</code>	Dense co-assignment limit passed to consensus_som() .
<code>keep_models</code>	Whether to retain fitted SOM and reference model objects.
<code>fail_fast</code>	Whether a model or consensus failure should stop the run.
<code>parallel</code>	Whether SOM ensemble members should use the current future plan; passed to fit_som_ensemble() .
<code>max_pairwise_comparisons</code>	Pairwise partition-comparison budget passed to partition_som() .

Value

A `som_workflow` object containing each evidence stream and explicit failure logs.

Examples

```
data <- simulate_som_scenario("clusters", n = 60, p = 4, seed = 6)
specification <- som_spec(c(3, 2), seeds = 1:2, rlen = 10, k = 2:3)
workflow <- run_som_workflow(
  data, specification, cross_models = c("kmeans", "ward")
)
summary(workflow)
```

simulate_som_scenario *Simulate known-structure data for SOM validation*

Description

Simulate known-structure data for SOM validation

Usage

```
simulate_som_scenario(
  scenario = c("gradient", "clusters", "overlap", "grouped_pseudoreplication",
    "multilayer_conflict"),
  n = 180L,
  p = 6L,
  seed = 1L,
  noise_sd = 1,
  class_probs = c(1, 1, 1),
  n_groups = NULL,
  group_icc = 0.7,
  n_domains = 3L,
  domain_shift = 0,
  missing_rate = 0,
  missing_mechanism = c("none", "mcar", "domain"),
  id = NULL
)
```

Arguments

scenario	One of "gradient", "clusters", "overlap", "grouped_pseudoreplication" or "multilayer_conflict".
n	Number of samples.
p	Number of variables per primary layer.
seed	Random seed.
noise_sd	Within-structure residual standard deviation.
class_probs	Three positive class proportions for simulations with genuine discrete classes. They are normalized internally. Ignored for a continuous gradient.
n_groups	Number of sampling groups for "grouped_pseudoreplication". The default is approximately one group per six observations, with a minimum of 10 groups.

group_icc	Target intraclass correlation induced by the simulated group effect. Used only for "grouped_pseudoreplication".
n_domains	Number of balanced sampling domains. A gradient always has domains spanning its latent coordinate. Other scenarios receive domains when domain_shift > 0 or missing_mechanism = "domain".
domain_shift	Magnitude of an additive domain-specific shift applied to the first two variables. Zero creates no additional domain shift.
missing_rate	Expected fraction of entries set to missing.
missing_mechanism	One of "none", "mcar" or "domain". Domain missingness is concentrated in the final sampling domain.
id	Optional stable sample identifiers. Supply these when independently constructed simulated objects will be compared in a sensitivity workflow.

Value

A som_data object with additional truth and simulation_spec components. Known latent variables and labels are never included in the training layers. Discrete labels are also copied to external_label for post hoc agreement assessment.

Lifecycle

Experimental. Existing scenario names are retained within the 1.x series, while new scenarios and scenario-specific controls may be added in minor releases.

Examples

```
simulated <- simulate_som_scenario(
  "gradient", n = 60, p = 4, seed = 1, domain_shift = 0.5
)
simulated
```

som_data

Construct a design-explicit SOM data object

Description

Construct a design-explicit SOM data object

Usage

```
som_data(
  x = NULL,
  layers = NULL,
  id = NULL,
  group = NULL,
```

```

    time = NULL,
    domain = NULL,
    weight = NULL,
    external_label = NULL
  )

```

Arguments

<code>x</code>	A numeric matrix or data frame for a single-layer analysis.
<code>layers</code>	A named list of numeric matrices or data frames. Use either <code>x</code> or <code>layers</code> , not both.
<code>id</code>	Optional unique sample identifiers.
<code>group</code>	Optional sampling-group identifiers.
<code>time</code>	Optional time values retained as metadata.
<code>domain</code>	Optional monitoring-domain identifiers.
<code>weight</code>	Optional non-negative survey or summary weights. These weights are retained only as metadata in the current release. They are not used in SOM training, evidence metrics or design summaries.
<code>external_label</code>	Optional external ecological labels. They are retained for post hoc assessment and never added to the training layers. A fully named vector is aligned to <code>id</code> ; partial, duplicate or unmatched label IDs are rejected. An unnamed vector is interpreted positionally.

Value

An object of class `som_data`. Named multi-layer inputs are aligned to the first layer by row name when all layers provide row names; incompatible or partly named layers are rejected. An explicit `id` vector is always interpreted positionally against the first layer and never changes its row order. If no layers have usable row names, their existing row order is treated as authoritative. The scalar `id_source` records whether sample identifiers were supplied, taken from explicit non-positional row names or generated locally. Stable identifiers are required to compare different data-coverage scenarios.

Examples

```

x <- data.frame(temperature = 10:14, oxygen = c(9, 8, 8, 7, 6))
data <- som_data(x, id = paste0("sample_", seq_len(nrow(x))))
data

```

`som_gate`

Define explicit evidence gates for a hard SOM partition

Description

No universal thresholds are supplied. A gate records the analyst's prespecified requirements.

Usage

```
som_gate(
  max_topographic_error = NULL,
  max_empty_unit_rate = NULL,
  min_median_ari = NULL,
  min_median_ami = NULL,
  min_pairwise_coverage = NULL,
  min_cluster_jaccard = NULL,
  min_consensus_coverage = NULL,
  min_replicated_coverage = NULL,
  min_membership_support = NULL,
  max_assignment_entropy = NULL,
  min_cross_model_ari = NULL,
  min_cross_model_methods = NULL,
  min_success_rate = NULL
)
```

Arguments

max_topographic_error
Optional maximum median topographic error.

max_empty_unit_rate
Optional maximum median empty-unit rate.

min_median_ari Optional minimum median pairwise ARI.

min_median_ami Optional minimum median pairwise AMI.

min_pairwise_coverage
Optional minimum median proportion of samples jointly assigned in pairwise partition comparisons.

min_cluster_jaccard
Optional minimum across cluster-level median Jaccard values.

min_consensus_coverage
Optional minimum proportion of samples with an identifiable consensus label.

min_replicated_coverage
Optional minimum proportion of samples assigned by at least two partitions, for which support and entropy are estimable.

min_membership_support
Optional minimum median sample membership support.

max_assignment_entropy
Optional maximum median normalized assignment entropy.

min_cross_model_ari
Optional minimum of the method-specific median SOM-to-reference ARI values. Each requested reference method must have an evaluable result; pooled comparisons are not used for this gate.

min_cross_model_methods
Optional minimum number of reference methods represented in the comparison.

min_success_rate
Optional minimum model-fit success rate.

Value

A som_gate object.

Examples

```
som_gate(min_success_rate = 0.95, min_cluster_jaccard = 0.75)
```

som_preprocess

Define leakage-safe SOM preprocessing

Description

Define leakage-safe SOM preprocessing

Usage

```
som_preprocess(
  transform = "identity",
  center = TRUE,
  scale = TRUE,
  zero_replacement = NULL
)
```

Arguments

transform	One of "identity", "log", "log1p", "sqrt", "hellinger" or "clr". "log" denotes the natural logarithm. A vector can assign "identity", "log1p", "log" or "sqrt" by column; name that vector to protect against column-order changes. Hellinger and CLR are whole-matrix transformations and must be specified alone.
center	Whether to subtract training-set column means.
scale	Whether to divide by training-set column standard deviations.
zero_replacement	Positive replacement required when transform = "clr" and zeros are present.

Value

A preprocessing specification of class som_preprocess.

Examples

```
som_preprocess(transform = c("log", "identity"))
```

som_resamples	<i>Construct design-aware resamples</i>
---------------	---

Description

Construct design-aware resamples

Usage

```
som_resamples(
  data,
  method = c("full", "subsample", "group_subsample", "block_subsample",
    "leave_domain_out", "custom"),
  repeats = 50L,
  prop = 0.8,
  seed = 1L,
  unit = "group",
  domain = "domain",
  splits = NULL
)
```

Arguments

data	A som_data object.
method	One of "full", "subsample", "group_subsample", "block_subsample", "leave_domain_out" or "custom".
repeats	Number of repeated subsamples.
prop	Proportion of rows or sampling units retained for analysis.
seed	Random seed used to generate all splits.
unit	Metadata column name or vector defining groups or predefined blocks. block_subsample does not infer blocks from row order.
domain	Metadata column name or vector defining transfer domains.
splits	For method = "custom", a list containing analysis and optional assessment row indices.

Value

A som_resamples object. The object records the number of distinct analysis sets in `n_unique_analysis_splits` and identifies repeated sets in `duplicate_analysis_splits`.

Examples

```
x <- matrix(seq_len(120), nrow = 30)
data <- som_data(x, group = rep(paste0("site_", 1:10), each = 3))
som_resamples(data, method = "group_subsample", repeats = 2, seed = 1)
```

som_spec

*Specify a reproducible SOM ensemble***Description**

Specify a reproducible SOM ensemble

Usage

```
som_spec(
  grids = data.frame(xdim = 7L, ydim = 5L),
  seeds = 1:10,
  rlen = 500L,
  alpha = c(0.05, 0.01),
  radius = NULL,
  topology = c("hexagonal", "rectangular"),
  neighbourhood = c("gaussian", "bubble"),
  toroidal = FALSE,
  mode = c("batch", "online", "pbatch"),
  k = 2:8,
  max_na_fraction = 0,
  layer_weights = NULL,
  normalize_layers = TRUE,
  cores = 1L
)
```

Arguments

grids	A two-column data frame with xdim and ydim, a numeric vector of length two for one grid, or a list of such vectors.
seeds	Integer random seeds.
rlen	Number of training iterations.
alpha	Initial and final learning rates.
radius	Optional initial and final neighbourhood radii.
topology	Rectangular or hexagonal grid.
neighbourhood	Bubble or Gaussian neighbourhood.
toroidal	Whether the grid is toroidal.
mode	Training mode supported by kohonen::supersom() .
k	Candidate numbers of hard partitions retained for later auditing.
max_na_fraction	Maximum allowed fraction of missing variables per layer and sample passed to kohonen.
layer_weights	Optional non-negative scientific weights, one per layer.

normalize_layers	Whether to divide each scientific layer weight by its training-set mean squared Euclidean distance. The package uses the variance identity for a deterministic, leakage-safe estimate and applies the same effective weights to SOM and cross-model fits.
cores	Number of cores used by kohonen in parallel batch mode.

Value

A som_spec object.

Examples

```
specification <- som_spec(
  grids = list(c(4, 3), c(5, 4)), seeds = 1:2, rlen = 50, k = 2:4
)
expand_som_spec(specification)
```

summary.som_workflow *Summarize workflow quality assurance and model outcomes*

Description

Summarize workflow quality assurance and model outcomes

Usage

```
## S3 method for class 'som_workflow'
summary(object, ...)
```

Arguments

object	A som_workflow object.
...	Reserved for future methods.

Value

A summary.som_workflow object containing fit, consensus, cross-model and provenance summaries plus the original failure tables.

upgrade_som_object	<i>Upgrade a persisted SOMEvidence object contract</i>
--------------------	--

Description

Upgrades a supported public SOMEvidence object to the current structural contract using only deterministic, non-analytical migrations. The function never refits a model or fabricates scientific evidence. If a required field cannot be reconstructed, it asks the user to recompute the affected result.

Usage

```
upgrade_som_object(x)
```

Arguments

x	A persisted public som_* object returned by SOMEvidence.
---	--

Details

Contract versions describe object structure, not the package or algorithm version that generated an analysis. Retain som_workflow\$provenance and sessionInfo() for that purpose.

Value

A validated object under the current contract. The input is not modified in place. Repeated upgrades are idempotent.

Examples

```
data <- som_data(matrix(seq_len(18), nrow = 6))
legacy <- data
attr(legacy, "som_contract_version") <- NULL
upgrade_som_object(legacy)
```

Index

`assess_defensibility`, 3
`assess_defensibility()`, 5, 15
`audit_som`, 4
`audit_som_representation`, 4
`audit_transfer`, 6

`compare_cross_models`, 6
`consensus_som`, 7
`consensus_som()`, 22

`evaluate_external_labels`, 8
`expand_som_spec`, 9

`fit_cross_models`, 10
`fit_cross_models()`, 22
`fit_som_ensemble`, 11
`fit_som_ensemble()`, 22

`kohonen::supersom()`, 29

`launch_som_app`, 12

`map_som_ensemble`, 13
`mclust::Mclust()`, 11

`open_data_registry`, 13

`pareto_candidates`, 14
`partition_som`, 15
`partition_som()`, 22
`plot.som_audit`, 16
`plot.som_consensus`, 16
`plot.som_cross_comparison`, 17
`plot.som_newdata_mapping`, 17
`plot.som_partitions`, 18
`plot.som_sensitivity`, 18
`plot.som_transfer_audit`, 19
`plot_som_plane`, 19

`run_som_sensitivity`, 20
`run_som_workflow`, 21

`run_som_workflow()`, 20

`shiny::runApp()`, 12
`simulate_som_scenario`, 23
`som_data`, 24
`som_data()`, 9
`som_gate`, 25
`som_preprocess`, 27
`som_resamples`, 28
`som_spec`, 29
`summary.som_workflow`, 30

`upgrade_som_object`, 31