Package 'palimpsestr'

July 29, 2026

Type Package

Title Probabilistic Decomposition of Archaeological Palimpsests

Version 0.24.1

Description Probabilistic framework for the analysis of archaeological palimpsests based on the Stratigraphic Entanglement Field (SEF). Integrates spatial proximity, stratigraphic depth, chronological overlap, and cultural similarity to estimate latent depositional phases via diagonal Gaussian mixture Expectation-Maximisation (EM). Provides the Stratigraphic Entanglement Index (SEI), Excavation Stratigraphic Energy (ESE), and Palimpsest Dissolution Index (PDI) for quantifying depositional coherence, detecting intrusive finds, and measuring palimpsest formation. Includes simulation, diagnostics, phase-count selection, publication-quality plots, and Geographic Information System (GIS) export via 'sf'. Methods are described in Cocca (2026) <https://github.com/enzococca/palimpsestr>.

URL https://github.com/enzococca/palimpsestr

BugReports https://github.com/enzococca/palimpsestr/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports stats, utils, graphics, grDevices

Suggests testthat (>= 3.0.0), knitr, rmarkdown, tinytex, sf, ggplot2, viridis, ggrepel, plotly, rlang, DBI, RSQLite, RPostgres, shiny, shinydashboard, DT, rcarbon (>= 1.5.0), oxcAAR

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Config/roxygen2/version 8.0.0

Author Enzo Cocca [aut, cre] (ORCID: <https://orcid.org/0000-0002-8096-2810>)

Maintainer Enzo Cocca <enzo.ccc@gmail.com>

Repository CRAN

Date/Publication 2026-07-29 13:30:02 UTC

Contents

`adjusted_rand_index` *Adjusted Rand Index*

Description

Compares estimated phase assignments against known true labels using the Adjusted Rand Index (Hubert and Arabie, 1985). Values near 1 indicate perfect agreement; values near 0 indicate random agreement.

Usage

```
adjusted_rand_index(object, true_labels)
```

Arguments

`object`	A `sef_fit` object, or an integer vector of predicted labels.
`true_labels`	Integer vector of known phase assignments.

Value

A single numeric value in $[-1, 1]$.

See Also

confusion_matrix, fit_sef

Other validation: bootstrap_sef(), confusion_matrix(), cv_sef(), optimize_weights()

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1, mixing = 0.05)
fit <- fit_sef(x, k = 3, seed = 1)
adjusted_rand_index(fit, x$true_phase)
```

---

archaeo_sim *Simulate an archaeological palimpsest dataset*

Description

Generates a synthetic excavation dataset with known latent phases, controlled spatial clustering, and configurable inter-phase mixing.

Usage

```
archaeo_sim(n = 150, k = 3, seed = NULL, mixing = 0.08)
```

Arguments

n	Number of observations (finds).
k	Number of latent depositional phases.
seed	Optional random seed for reproducibility.
mixing	Proportion of observations to perturb spatially and taphonomically, simulating post-depositional disturbance (0–1).

Value

A data.frame with columns: id, x, y, z, context, date_min, date_max, class, taf_score, true_phase. The date bounds date_min and date_max are returned as integer-valued years (via floor()), consistent with archaeological dating conventions.

See Also

fit_sef for fitting the SEF model to the output.

Examples

```
easy <- archaeo_sim(n = 100, k = 3, mixing = 0.05, seed = 1)
table(easy$true_phase)

hard <- archaeo_sim(n = 200, k = 4, mixing = 0.50, seed = 1)
table(hard$true_phase)
```

as_phase_table *Extract phase probability table*

Description

Returns a data.frame combining dominant phase assignments, membership probabilities, entropy, local SEI, and energy.

Usage

```
as_phase_table(object)
```

Arguments

object	A sef_fit object.

Value

A data.frame with one row per find.

See Also

predict_phase, phase_diagnostic_table

Other diagnostics: detect_intrusions(), ese(), pdi(), phase_composition(), phase_diagnostic_table(), predict_phase(), recommend_setup(), type_longevity()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
head(as_phase_table(fit))
```

as_plotly *Convert a ggplot to an interactive plotly widget*

Description

Wraps ggplotly with archaeological tooltips showing find ID, context, phase probability, dating, class, entropy, and energy.

Usage

```
as_plotly(gg, tooltip = "text", ...)
```

Arguments

gg	A ggplot object produced by any gg_* function.
tooltip	Character vector of aesthetics to show. Defaults to "text" which displays the enriched archaeological tooltip.
...	Additional arguments passed to ggplotly.

Value

A plotly htmlwidget object.

See Also

gg_phasefield, gg_entropy, gg_energy, gg_intrusions

Other plotting: gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
fit <- fit_sef(x, k = 3)
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("plotly", quietly = TRUE)) {
  p <- as_plotly(gg_phasefield(fit))
}
```

---

as_sf_links *Export high-SEI links as an sf LINESTRING layer*

Description

Extracts the strongest pairwise SEI connections as sf line geometries.

Usage

```
as_sf_links(object, quantile_threshold = 0.9, crs = NA_integer_)
```

Arguments

object	A sef_fit object.
quantile_threshold	Quantile for retaining strongest links (default: 0.9).
crs	CRS for the output geometry.

Value

An sf object with columns from, to, sei, and geometry.

See Also

as_sf_phase, sei_matrix

Other GIS: as_sf_phase()

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  x <- archaeo_sim(n = 60, k = 2, seed = 1)
  fit <- fit_sef(x, k = 2)
  links <- as_sf_links(fit)
}
```

---

as_sf_phase *Convert a fitted model to an sf point layer*

Description

Creates an sf point object with phase assignments and diagnostics for use in QGIS or spatial analysis.

Usage

```
as_sf_phase(object, crs = NA_integer_, dims = c("XY", "XYZ"))
```

Arguments

object	A sef_fit object.
crs	CRS passed to st_as_sf.
dims	Either "XY" or "XYZ".

Value

An sf object.

See Also

as_sf_links, phase_diagnostic_table

Other GIS: as_sf_links()

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  x <- archaeo_sim(n = 60, k = 2, seed = 1)
  fit <- fit_sef(x, k = 2)
  pts <- as_sf_phase(fit)
}
```

`bootstrap_sef` *Bootstrap confidence intervals for SEF diagnostics*

Description

Resamples the data with replacement, refits the model, and computes the distribution of key statistics (PDI, mean entropy, mean energy, ARI if true labels provided). Returns percentile confidence intervals.

Usage

```
bootstrap_sef(
  object,
  n_boot = 100,
  conf = 0.95,
  true_labels = NULL,
  verbose = TRUE
)
```

Arguments

`object`	A `sef_fit` object (used to extract fitting parameters).
`n_boot`	Number of bootstrap replicates (default: 100).
`conf`	Confidence level (default: 0.95).
`true_labels`	Optional integer vector of true phase labels for ARI.
`verbose`	Print progress (default: TRUE).

Value

A data.frame with columns `statistic`, `estimate`, `lower`, `upper`, `se`.

See Also

`fit_sef`, `pdi`, `adjusted_rand_index`

Other validation: `adjusted_rand_index()`, `confusion_matrix()`, `cv_sef()`, `optimize_weights()`

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2, seed = 1)
bootstrap_sef(fit, n_boot = 20)
```

`chronology_from_oxcal` *Convert OxCal-calibrated dates to date_min/date_max columns*

Description

Adapter that takes either an `oxcAAR`-calibrated dates object or a generic data.frame of calibrated ranges and returns a data.frame compatible with the chronology columns expected by `fit_sef`. It is the OxCal counterpart of `chronology_from_rcarbon`.

Usage

```
chronology_from_oxcal(
  x,
  method = c("hpd", "median_iqr", "weighted_mean"),
  hpd = 0.95,
  ids = NULL,
  bce_negative = TRUE
)
```

Arguments

`x`	Either an object of class `oxcAARCalibratedDatesList` (from `oxcAAR::oxcalCalibrate()`) or a `data.frame` with numeric `start` and `end` columns (calendar years, BC negative) and an optional `id` column.
`method`	For the oxcAAR object input, one of `"hpd"` (default), `"median_iqr"`, or `"weighted_mean"`. Ignored (with a message) for data.frame input, which already supplies an interval.
`hpd`	Probability covered by the HPD region when `method = "hpd"` (default 0.95).
`ids`	Optional character identifiers; default `cal_1 ... cal_n`.
`bce_negative`	Logical. If TRUE (default), dates are returned in the BCE/CE convention with BCE negative; if FALSE, raw calBP (`1950 - year`).

Value

A data.frame with columns `id`, `date_min`, `date_max`, `date_mid`.

See Also

`chronology_from_rcarbon`, `fit_sef`

Other chronology: `chronology_from_rcarbon()`

Examples

```
d <- data.frame(id = c("a", "b"), start = c(-700, -50), end = c(-600, 100))
chronology_from_oxcal(d)
```

`chronology_from_rcarbon` *Convert calibrated radiocarbon dates to date_min/date_max columns*

Description

Adapter that takes an `rcarbon::CalDates` object (produced by `rcarbon::calibrate()`) and returns a `data.frame` compatible with the chronology columns expected by `fit_sef()`.

Usage

```
chronology_from_rcarbon(
  cal_dates,
  method = c("hpd", "median_iqr", "weighted_mean"),
  hpd = 0.95,
  ids = NULL,
  bce_negative = TRUE
)
```

Arguments

`cal_dates`	An object of class `CalDates` produced by `rcarbon::calibrate()`.
`method`	One of `"hpd"` (Highest Posterior Density region), `"median_iqr"` (weighted median plus 25\ or `"weighted_mean"` (mean +/- 2 SD weighted by density).
`hpd`	Numeric in (0, 1). Probability covered by the HPD region when `method = "hpd"` (default: 0.95).
`ids`	Optional character vector of length `length(cal_dates$grids)` used as `id`. Defaults to `paste0("cal_", seq_along(...))`.
`bce_negative`	Logical. If TRUE (default), dates are returned in the BCE/CE convention with BCE as negative numbers (calBP converted as `1950 - calBP`). If FALSE, raw calBP is returned.

Value

A data.frame with columns `id`, `date_min`, `date_max`, `date_mid`.

See Also

`fit_sef`

Other chronology: `chronology_from_oxcal()`

Examples

```
## Not run:
if (requireNamespace("rcarbon", quietly = TRUE)) {
  cal <- rcarbon::calibrate(x = c(2500, 2400), errors = c(30, 30))
  chronology_from_rcarbon(cal)
}

## End(Not run)
```

`compare_k` *Compare multiple candidate phase counts*

Description

Fits the SEF model for each value of K and returns a summary table with BIC, PDI, entropy, energy, and other diagnostics.

Usage

```
compare_k(data, k_values = 2:6, ...)
```

Arguments

`data`	Input data.frame.
`k_values`	Integer vector of candidate phase counts.
`...`	Additional arguments passed to `fit_sef`.

Details

For each candidate `K` the table reports several model-selection quantities. The **Bayesian Information Criterion** (`bic`, $-2 \log \hat{L} + d \log n$, where $\hat{L}$ is the fitted mixture likelihood, d the number of free parameters and n the number of finds; Schwarz 1978) balances goodness of fit against model complexity and is the primary selector — lower is better. The **Palimpsest Dissolution Index** (`pdi`; see `pdi`) is a 0–1 measure of how cleanly the phases separate, and `mean_entropy` is the mean Shannon entropy of the soft assignments. The **Integrated Completed Likelihood** (`icl`) augments the BIC with the classification entropy ($\mathrm{ICL} = \mathrm{BIC} + 2 \sum_i H_i$; Biernacki, Celeux & Govaert 2000), penalising overlapping phases so that it favours partitions with confident assignments. Statistical criteria should be combined with archaeological judgement: the BIC minimum is not necessarily the most interpretable partition.

Value

A data.frame with one row per K value.

References

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 461–464. doi:10.1214/aos/1176344136

Biernacki, C., Celeux, G., & Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7), 719–725. doi:10.1109/34.865189

See Also

`fit_sef`, `gg_compare_k`, `pdi`

Other fitting: `fit_sef()`, `reorder_phases()`, `sef_summary()`

Examples

```
x <- archaeo_sim(n = 100, k = 3, seed = 1)
ck <- compare_k(x, k_values = 2:4)
print(ck)
```

`confusion_matrix` *Confusion matrix between estimated and true phases*

Description

Cross-tabulates estimated phase assignments against known true labels. Phases are reordered to maximise diagonal agreement (Hungarian matching).

Usage

```
confusion_matrix(object, true_labels)
```

Arguments

`object`	A `sef_fit` object, or an integer vector of predicted labels.
`true_labels`	Integer vector of known phase assignments.

Value

A matrix with estimated phases as rows and true phases as columns.

See Also

`adjusted_rand_index`

Other validation: `adjusted_rand_index()`, `bootstrap_sef()`, `cv_sef()`, `optimize_weights()`

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1, mixing = 0.05)
fit <- fit_sef(x, k = 3, seed = 1)
confusion_matrix(fit, x$true_phase)
```

`cv_sef` *K-fold cross-validation for SEF model*

Description

Splits the data into folds, fits on training folds, and evaluates log-likelihood on the held-out fold. Useful for comparing different K values or weight configurations.

Usage

```
cv_sef(data, k_values = 2:6, n_folds = 5, seed = 1, ...)
```

Arguments

`data`	A data.frame with archaeological find data.
`k_values`	Integer vector of candidate phase counts.
`n_folds`	Number of cross-validation folds (default: 5).
`seed`	Random seed for fold assignment.
`...`	Additional arguments passed to `fit_sef`.

Value

A data.frame with columns `k`, `fold`, `train_loglik`, `test_loglik`, `train_pdi`. The `test_loglik` carries the same Jacobian correction as `optimize_weights`, so it is comparable across weight configurations (not only across k); with default weights the correction is zero.

See Also

`compare_k`, `fit_sef`

Other validation: `adjusted_rand_index()`, `bootstrap_sef()`, `confusion_matrix()`, `optimize_weights()`

Examples

```
x <- archaeo_sim(n = 100, k = 3, seed = 1)
cv <- cv_sef(x, k_values = 2:4, n_folds = 3)
# Mean test log-likelihood per K
aggregate(test_loglik ~ k, data = cv, FUN = mean)
```

demo_compressed *Demo dataset: compressed palimpsest*

Description

Simulated dataset with 250 artefacts, 4 phases, and 50% mixing. Represents a heavily disturbed deposit where phases are difficult to separate.

Usage

```
demo_compressed
```

Format

A data.frame with 250 rows and 10 columns. See demo_easy for column descriptions.

Examples

```
data(demo_compressed)

ck <- compare_k(demo_compressed, k_values = 2:6)
print(ck)
```

demo_easy *Demo dataset: well-separated phases*

Description

Simulated dataset with 150 artefacts, 3 phases, and 5% mixing. Represents a site where occupation phases are clearly distinguishable.

Usage

```
demo_easy
```

Format

A data.frame with 150 rows and 10 columns:

id Artefact identifier

x, y Planimetric coordinates (metres)

z Depth (metres)

context Stratigraphic unit label

date_min, date_max Chronological interval (CE)

class Cultural class (ceramic, lithic, bone, metal)

taf_score Taphonomic disturbance score (0-1)

true_phase Known phase assignment (for validation)

Examples

```
data(demo_easy)
fit <- fit_sef(demo_easy, k = 3)
plot_phasefield(fit)
```

`demo_moderate` *Demo dataset: moderate palimpsest*

Description

Simulated dataset with 200 artefacts, 3 phases, and 30% mixing. Represents a site with significant but resolvable depositional mixing.

Usage

```
demo_moderate
```

Format

A data.frame with 200 rows and 10 columns. See `demo_easy` for column descriptions.

Examples

```
data(demo_moderate)
fit <- fit_sef(demo_moderate, k = 3, tafonomy = "taf_score", context = "context")
summary(fit)
```

`detect_intrusions` *Detect potentially intrusive observations*

Description

Returns a per-find intrusion probability. When the model was fitted with `noise = TRUE`, `intrusion_prob` is the posterior probability of the uniform noise component (a genuine outlier probability). Otherwise it is the heuristic composite of rescaled entropy, energy, and inverse local SEI.

Usage

```
detect_intrusions(object, envelope = c(0.05, 0.95), intrusion_threshold = 0.5)
```

Arguments

object	A sef_fit object.
envelope	Numeric vector of two probabilities giving the quantiles of the other finds' dating bounds used as the leave-one-out unit envelope for the directional classification. The default c(0.05, 0.95) is robust to a single chronological outlier within the context; use c(0, 1) for the strict min/max envelope used prior to v0.14.0.
intrusion_threshold	Probability above which a find counts as flagged for the intrusion_type classification (default: 0.5).

Value

A data.frame with columns id, intrusion_prob, direction (factor with levels older_than_context, in_context, younger_than_context), and chrono_gap (numeric, signed offset in years). The data.frame also carries an intrusion_type factor combining the intrusion magnitude with the direction: not_flagged, residual (flagged and older-than-context), latent_feature (flagged and younger-than-context), or outlier_in_context.

See Also

gg_intrusions, fit_sef, type_longevity

Other diagnostics: as_phase_table(), ese(), pdi(), phase_composition(), phase_diagnostic_table(), predict_phase(), recommend_setup(), type_longevity()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
di <- detect_intrusions(fit)
head(di[order(di$intrusion_prob, decreasing = TRUE), ])
```

---

ese *Compute Excavation Stratigraphic Energy*

Description

Measures local depositional disruption for each find by summing weighted dissimilarities with neighbours.

Usage

```
ese(
  data,
  coords = c("x", "y", "z"),
  chrono = c("date_min", "date_max"),
  class_col = "class",
```

```
  beta = c(1, 1, 1, 1),
  neighbourhood = NULL
)
```

Arguments

data	Input data.frame.
coords	Character vector of coordinate column names.
chrono	Character vector with minimum and maximum dating columns.
class_col	Class column name.
beta	Numeric vector of length 4: weights for spatial, vertical, temporal, and class mismatch. Since v0.14.0 each component is normalised to $[0, 1]$ before weighting, so the energy is independent of the measurement unit of the coordinates and the betas express relative importance.
neighbourhood	Maximum XY distance for neighbour inclusion. When NULL, all observations contribute.

Value

A numeric vector of local energy values.

See Also

fit_sef, gg_energy

Other diagnostics: as_phase_table(), detect_intrusions(), pdi(), phase_composition(), phase_diagnostic_table(), predict_phase(), recommend_setup(), type_longevity()

Examples

```
x <- archaeo_sim(n = 30, k = 2, seed = 1)
e <- ese(x)
summary(e)
```

---

export_results *Export all results to files*

Description

Writes phase assignments, intrusion scores, US summary, and model summary to CSV files in the specified directory.

Usage

```
export_results(object, dir, format = "csv", prefix = "palimpsestr")
```

Arguments

object	A sef_fit object.
dir	Output directory (created if it does not exist). Must be specified explicitly; consider using tempdir() for temporary output.
format	Export format: "csv" (default).
prefix	File name prefix (default: "palimpsestr").

Value

Invisibly returns a character vector of written file paths.

See Also

us_summary_table, as_phase_table

Other export: phase_transition_matrix(), us_summary_table()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2, context = "context")
export_results(fit, dir = tempdir())
```

---

export_sef_report *Export a narrated SEF analysis report (PDF / DOCX)*

Description

Assembles a complete report from a fitted sef_fit: the interpretive narrative of report_sef, all applicable gg_* diagnostic plots, and diagnostic tables (US summary, phase-transition matrix, per-US phase assignments, model statistics). The report is rendered to PDF and/or DOCX via an RMarkdown template.

Usage

```
export_sef_report(
  fit,
  file,
  format = c("pdf", "docx"),
  lang = c("it", "en"),
  title = NULL,
  site = NULL,
  compare_k = NULL,
  intrusion_threshold = 0.5,
  quiet = TRUE
)
```

Arguments

fit	A fitted sef_fit object.
file	Output path. Any extension is stripped and replaced; sibling files (.pdf, .docx, .md, _figs/) are written next to it.
format	Character vector, any of "pdf" and "docx".
lang	Report language: "it" (default) or "en".
title	Report title (a sensible localized default is used when NULL).
site	Optional site name printed in the header.
compare_k	Optional result of compare_k; when supplied a K-comparison plot is included.
intrusion_threshold	Posterior threshold for flagging intrusions (default 0.5).
quiet	Passed to rmarkdown::render (default TRUE).

Details

A markdown narrative sidecar (<file>.md) is **always** written (used, for example, by the QGIS results panel). When pandoc is unavailable (as can happen when R is launched outside RStudio), the function degrades gracefully to that markdown narrative plus the figures saved as PNG, and tells you how to enable full rendering.

Value

(Invisibly) a character vector of the files written.

See Also

report_sef, fit_sef

---

fit_sef *Fit the Stratigraphic Entanglement Field model*

Description

Estimates latent depositional phases from spatial, stratigraphic, chronological, and cultural evidence using diagonal Gaussian mixture EM.

Usage

```
fit_sef(
  data,
  coords = c("x", "y", "z"),
  chrono = c("date_min", "date_max"),
  class = "class",
  tafonomy = NULL,
  context = NULL,
```

```
  harris = NULL,
  k = 3,
  weights = c(ws = 1, wz = 1, wt = 1, wc = 1),
  seed = 1,
  em_iter = 100,
  em_tol = 1e-05,
  n_init = 5,
  chrono_precision = FALSE,
  taf_as_feature = FALSE,
  residuality = FALSE,
  class_scale = FALSE,
  subclass = NULL,
  var_structure = c("diagonal", "spherical"),
  class_model = c("multinomial", "gaussian"),
  class_smoothing = 1,
  chrono_uncertainty = FALSE,
  strat_dynamic = FALSE,
  strat_beta = 1,
  noise = FALSE,
  noise_prior = 0.05
)
```

Arguments

data	A data.frame with archaeological find data.
coords	Character vector of length 3 naming the x, y, z coordinate columns.
chrono	Character vector of length 2 naming the min and max dating columns.
class	Character scalar naming the material class column.
tafonomy	Optional column name for taphonomic disturbance scores (0–1).
context	Optional column name for stratigraphic unit labels.
harris	Optional $n \times n$ matrix of pairwise stratigraphic penalties.
k	Integer number of phases to estimate.
weights	Named numeric vector with components ws, wz, wt, wc (all strictly positive). Since v0.14.0 the weights scale the standardised feature dimensions (ws: planar coordinates, wz: depth, wt: chronology-derived features, wc: class block) and therefore enter the mixture likelihood, in addition to weighting the SEI components. optimize_weights can select them by cross-validation.
seed	Random seed for reproducibility.
em_iter	Maximum number of EM iterations (default: 100).
em_tol	Convergence tolerance on the log-likelihood.
n_init	Number of random initialisations. The run with the highest EM objective is retained (default: 5). The EM objective is multimodal, so a single initialisation risks a poor local optimum.
chrono_precision	Logical. If TRUE, add 1/tspan as a feature giving higher weight to precisely dated finds (default: FALSE).

`taf_as_feature` Logical. If TRUE and `tafonomy` is provided, include taf_score as an additional feature dimension (default: FALSE).

`residuality` Logical. If TRUE and `context` is provided, add a residuality score measuring chronological mismatch between each find and its context mean (default: FALSE).

`class_scale` Logical. If TRUE, scale one-hot class columns by `1/sqrt(n_classes)` so their total energy is comparable to a single numeric feature (default: FALSE).

`subclass` Optional column name for sub-class labels. If provided, used instead of `class` for one-hot encoding in the feature matrix.

`var_structure` Either `"diagonal"` (default) for free per-dimension component variances, or `"spherical"` for one shared variance per component in the weighted feature space. Free diagonal variances absorb any rescaling of the feature columns, so the domain `weights` influence a diagonal fit only through the initialisation; under `"spherical"` the weights define the relative precision of the dimensions, are identifiable, and can be selected by `optimize_weights`.

`class_model` Either `"multinomial"` (default) or `"gaussian"`. Under `"multinomial"` the class (or `subclass`) label is modelled as a per-phase categorical distribution, giving a Gaussian-times-multinomial mixed-type mixture. Under `"gaussian"` (the behaviour up to v0.14.0) the class label is one-hot encoded into the Gaussian feature block; because zero/one dummies make finds of different classes almost perfectly separable, that model tends to produce over-confident assignments (entropy near 0, PDI near 1). The multinomial model is recommended; the Gaussian model is retained for backward compatibility.

`class_smoothing`

Non-negative Dirichlet pseudo-count for the categorical class probabilities under `class_model = "multinomial"` (default: 1). The pseudo-count is shrunk towards the global class frequencies, keeping every per-phase class probability strictly positive; larger values pull the phase class profiles towards the overall distribution, `0` removes the shrinkage entirely.

`chrono_uncertainty`

Logical (default `FALSE`). If `TRUE`, the dating interval width of each find is propagated into the likelihood: the mid-date is treated as observed with a measurement variance `(date_max - date_min)^2 / 12` (a uniform prior over the interval), added to the temporal dimension's component variance in the E-step and removed from the component variance estimate by moment deconvolution in the M-step. Finds with wide dating ranges then pull the temporal centroids less and carry their chronological uncertainty into the phase probabilities. Unlike `chrono_precision` (which adds `1/tspan` as an extra feature), this is a model-based treatment of the dating uncertainty.

`strat_dynamic` Logical (default `FALSE`). If `TRUE` and `context` is provided, the stratigraphic constraint is applied as a Neighborhood-EM / hidden-Markov-random-field term recomputed from the current phase posteriors at every E-step, instead of the static penalty frozen from the initial clustering. Each find is rewarded for sharing the phase of its stratigraphic-unit neighbours, encouraging units to stay coherent as the fit evolves (Ambroise & Govaert, 1997). Any `harris` penalty continues to be applied statically.

strat_beta	Non-negative strength of the dynamic stratigraphic field (default 1); only used when strat_dynamic = TRUE. 0 disables the field.
noise	Logical (default FALSE). If TRUE, a uniform background ("noise") component is added to the mixture (Fraley & Raftery, 1998). Finds that fit no Gaussian phase accumulate posterior probability on it, so the fit stores a noise_prob vector that is a genuine posterior probability of being an outlier/intrusion (used by detect_intrusions in place of the heuristic composite score), and extreme finds are kept out of the phase centroid and variance estimates. The reported phase_prob is then the distribution over phases conditional on a find not being noise.
noise_prior	Initial mixing weight of the noise component, strictly between 0 and 1 (default 0.05); only used when noise = TRUE.

Value

An S3 object of class sef_fit.

See Also

archaeo_sim, compare_k, pdi, detect_intrusions

Other fitting: compare_k(), reorder_phases(), sef_summary()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
print(fit)


x <- archaeo_sim(n = 150, k = 3, seed = 42)
fit <- fit_sef(x, k = 3, tafonomy = "taf_score", context = "context")
summary(fit)
```

---

gg_bootstrap *Bootstrap confidence interval plot*

Description

Visualises bootstrap confidence intervals for key SEF diagnostics (PDI, entropy, energy, log-likelihood, and optionally ARI).

Usage

```
gg_bootstrap(bs_result)
```

Arguments

bs_result	Data.frame returned by bootstrap_sef.

Value

A ggplot object.

See Also

bootstrap_sef

Other plotting: as_plotly(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

---

gg_compare_k *Phase count selection diagnostics*

Description

Three-panel plot showing BIC, PDI, and Mean Entropy for different K values.

Usage

```
gg_compare_k(ck)
```

Arguments

ck	Data.frame from compare_k().

Value

A ggplot object.

See Also

compare_k

Other plotting: as_plotly(), gg_bootstrap(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

gg_confusion *Confusion matrix heatmap*

Description

Plots a heatmap of the confusion matrix between estimated and known true phase assignments.

Usage

```
gg_confusion(object, true_labels)
```

Arguments

object	A sef_fit object.
true_labels	Integer vector of known true phase assignments.

Value

A ggplot object.

See Also

confusion_matrix, adjusted_rand_index

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  x <- archaeo_sim(n = 80, k = 3, seed = 1, mixing = 0.05)
  fit <- fit_sef(x, k = 3, seed = 1)
  gg_confusion(fit, x$true_phase)
}
```

gg_convergence *EM convergence trace*

Description

Plots the log-likelihood at each EM iteration to verify convergence.

Usage

```
gg_convergence(object)
```

Arguments

object	A sef_fit object.

Value

A ggplot object.

See Also

fit_sef

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  x <- archaeo_sim(n = 80, k = 3, seed = 1)
  fit <- fit_sef(x, k = 3)
  gg_convergence(fit)
}
```

gg_cv *Cross-validation diagnostic plot*

Description

Plots mean test and training log-likelihood across K values from cv_sef output.

Usage

```
gg_cv(cv_result)
```

Arguments

cv_result	Data.frame returned by cv_sef.

Value

A ggplot object.

See Also

cv_sef, gg_compare_k

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

---

gg_direction *Directional intrusion plot*

Description

Diverging lollipop of the signed chronological offset (chrono_gap) for finds classified as residual (*older-than-context*) or latent (*younger-than-context*) by detect_intrusions.

Usage

```
gg_direction(object, top_n = 20)
```

Arguments

object	A sef_fit object.
top_n	Maximum number of finds to show, by largest absolute gap (default: 20).

Value

A ggplot object. When every dated find is in context (e.g. unit-tied chronology) an explanatory placeholder plot is returned.

See Also

detect_intrusions, gg_intrusions

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2, context = "context")
gg_direction(fit)
```

---

gg_energy *Excavation Stratigraphic Energy map*

Description

Shows ESE values across space. High energy indicates zones where neighbouring artefacts are dissimilar (depositional disruption).

Usage

```
gg_energy(object, xlabel = "Easting (m)", ylabel = "Northing (m)")
```

Arguments

object	A sef_fit object.
xlabel, ylabel	Axis labels.

Value

A ggplot object.

See Also

gg_entropy, as_plotly for interactive version.

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

---

gg_entropy *Spatial entropy map*

Description

Shows Shannon entropy of phase probabilities across the excavation area. High entropy = uncertain phase assignment (possible palimpsest zone).

Usage

```
gg_entropy(object, xlabel = "Easting (m)", ylabel = "Northing (m)")
```

Arguments

object	A sef_fit object.
xlabel, ylabel	Axis labels.

Value

A ggplot object.

See Also

plot_entropy for base R version, as_plotly for interactive version.

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

---

gg_intrusions *Intrusion probability map*

Description

Maps the probability that each find is an intrusion (displaced or redeposited). Top suspects are circled and labelled.

Usage

```
gg_intrusions(
  object,
  top_n = 5,
  xlabel = "Easting (m)",
  ylabel = "Northing (m)"
)
```

Arguments

object	A sef_fit object.
top_n	Number of top intrusions to label.
xlabel, ylabel	Axis labels.

Value

A ggplot object.

See Also

detect_intrusions, as_plotly for interactive version.

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

---

gg_longevity *Gantt-style plot of type longevity*

Description

Visualises the output of type_longevity() as a horizontal Gantt chart: one segment per class spanning longevity_min to longevity_max, ordered by longevity_min, coloured by dominant_phase.

Usage

```
gg_longevity(longevity_table, fit = NULL)
```

Arguments

longevity_table	A data.frame as returned by type_longevity().
fit	Optional sef_fit to harmonise the colour palette with other phase plots. Currently used only to extract k for the factor levels of dominant_phase.

Value

A ggplot object.

See Also

type_longevity

Examples

```
x <- archaeo_sim(n = 90, k = 3, seed = 1)
fit <- fit_sef(x, k = 3, context = "context")
tl <- type_longevity(fit)
if (requireNamespace("ggplot2", quietly = TRUE)) gg_longevity(tl, fit)
```

gg_map *Overlay SEF results on excavation plan geometries*

Description

Displays SEF analysis results directly on the excavation plan. Polygons represent stratigraphic units; points represent individual finds.

Usage

```
gg_map(
  object,
  geometries,
  layer = c("phase", "entropy", "energy", "intrusion"),
  show_labels = TRUE,
  show_points = TRUE,
  xlabel = NULL,
  ylabel = NULL
)
```

Arguments

object	A sef_fit object.
geometries	An sf object with excavation plan polygons. Must contain a us column matching context values in the data.
layer	What to display: "phase", "entropy", "energy", or "intrusion".
show_labels	Show US labels on polygons.
show_points	Overlay individual finds as points.
xlabel, ylabel	Axis labels (default: NULL = use CRS units).

Value

A ggplot object.

See Also

load_geometries, as_sf_phase

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

gg_outliers *Intrusion ranking plot*

Description

Non-spatial ranking of the per-find intrusion probability (complements gg_intrusions, which is the map). Uses the noise-component posterior when the model was fitted with noise = TRUE, otherwise the heuristic composite score.

Usage

```
gg_outliers(object, threshold = 0.5, top_n = 20)
```

Arguments

object	A sef_fit object.
threshold	Reference probability above which finds are highlighted (default: 0.5).
top_n	Number of highest-scoring finds to show (default: 20).

Value

A ggplot object.

See Also

detect_intrusions, gg_intrusions

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2, noise = TRUE)
gg_outliers(fit)
```

gg_phasefield *Phase assignment map*

Description

Plots artefact positions coloured by dominant phase, with point size proportional to assignment confidence.

Usage

```
gg_phasefield(object, xlabel = "Easting (m)", ylabel = "Northing (m)")
```

Arguments

object	A sef_fit object.
xlabel, ylabel	Axis labels (default: "Easting (m)" / "Northing (m)").

Value

A ggplot object.

See Also

plot_phasefield for base R version, as_plotly for interactive version.

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

gg_phase_composition *Per-phase class composition plot*

Description

Visualises the estimated per-phase categorical class profile (object$cat_prob) from a multinomial-class fit.

Usage

```
gg_phase_composition(object, type = c("heatmap", "bar"), top_n = NULL)
```

Arguments

object	A sef_fit fitted with class_model = "multinomial".
type	Either "heatmap" (default; phase x class tiles filled by probability) or "bar" (stacked bars per phase).
top_n	For type = "bar", show only the top_n classes by mean probability and collapse the rest into "other" (default: all).

Value

A ggplot object.

See Also

phase_composition, fit_sef

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
fit <- fit_sef(x, k = 3)
gg_phase_composition(fit)
```

---

gg_phase_profile *Vertical phase profile*

Description

Plots finds along the depth (z) axis, coloured by phase assignment, to visualise the stratigraphic ordering of phases.

Usage

```
gg_phase_profile(object)
```

Arguments

object	A sef_fit object.

Value

A ggplot object.

See Also

phase_transition_matrix

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  x <- archaeo_sim(n = 80, k = 3, seed = 1)
  fit <- fit_sef(x, k = 3)
  gg_phase_profile(fit)
}
```

---

gg_unit_coherence *Within-unit phase coherence plot*

Description

Shows, per stratigraphic unit, whether the finds were assigned to a single phase (coherent) or split across several (split). A coherent assignment is expected because a unit is one depositional event. An optional second fit (e.g. a `class_model = "gaussian"` fit) is shown side by side.

Usage

```
gg_unit_coherence(object, compare = NULL)
```

Arguments

`object`	A `sef_fit` fitted with a `context` column.
`compare`	Optional second `sef_fit` over the same data, shown alongside `object`.

Value

A ggplot object.

See Also

fit_sef, us_summary_table

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
fit <- fit_sef(x, k = 3, context = "context")
gg_unit_coherence(fit)
```

`gg_weights` *Weight sensitivity heatmap*

Description

Plots the mean test log-likelihood across weight configurations from `optimize_weights` output.

Usage

```
gg_weights(opt_result)
```

Arguments

`opt_result`	List returned by `optimize_weights`.

Value

A ggplot object.

See Also

`optimize_weights`

Other plotting: `as_plotly()`, `gg_bootstrap()`, `gg_compare_k()`, `gg_confusion()`, `gg_convergence()`, `gg_cv()`, `gg_direction()`, `gg_energy()`, `gg_entropy()`, `gg_intrusions()`, `gg_map()`, `gg_outliers()`, `gg_phase_composition()`, `gg_phase_profile()`, `gg_phasefield()`, `gg_unit_coherence()`, `plot_energy()`, `plot_entropy()`, `plot_phasefield()`, `plot_sei_profile()`

`harris_from_contexts` *Generate stratigraphic penalty matrix from context depth ordering*

Description

Infers vertical ordering between stratigraphic units from the mean depth of finds in each context, and builds a penalty matrix that discourages finds from different vertical zones being assigned to the same phase.

Usage

```
harris_from_contexts(
  data,
  z_col = "z",
  context_col = "context",
  penalty_scale = 0.5,
  exclude_contexts = NULL
)
```

Arguments

`data`	A data.frame with find data.
`z_col`	Name of the depth column.
`context_col`	Name of the context column.
`penalty_scale`	Penalty magnitude for cross-context assignments.
`exclude_contexts`	Optional character vector of context labels that are exempt from the verticality penalty (e.g. fills of cuts, or multi-period contexts where depth does not track chronology). Finds in these contexts receive a zero cross-context penalty, so they neither impose nor are subject to the depth-rank constraint. Defaults to `NULL` (no exemptions), preserving the original behaviour.

Value

An $n \times n$ symmetric penalty matrix.

Verticality assumption

This function encodes a *verticality rule*: it assumes that the mean depth of a context is a proxy for its relative chronological position (deeper = earlier) and penalises same-phase assignment of finds whose contexts differ in depth rank. The assumption holds for sub-horizontally stratified deposits but is **frequently violated** in real excavations — for inclined deposits and slope collapses, and especially for the fills of cuts (pits, ditches, post-holes) and for single contexts that contain material from several chronological periods. In such cases the depth-rank penalty conflates genuinely problematic configurations with perfectly normal archaeological features. Two escape hatches are provided: pass the affected contexts to `exclude_contexts` to exempt them from the penalty, or skip this helper entirely and supply an explicitly recorded Harris matrix via `read_harris`.

See Also

`fit_sef`, `read_harris`

Other harris: `read_harris()`, `validate_phases_harris()`

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
H <- harris_from_contexts(x, z_col = "z", context_col = "context")
dim(H)
```

```
# Exempt a known multi-period fill from the verticality rule:
H2 <- harris_from_contexts(x, exclude_contexts = "SU_3")
```

---

launch_app *Launch the palimpsestr Shiny Dashboard*

Description

Opens an interactive Shiny application for loading data, fitting the SEF model, and exploring results through plots, tables, and maps.

Usage

```
launch_app()
```

Details

The app requires the following packages (installed but not imported by palimpsestr): `shiny`, `shinydashboard`, `DT`. For interactive plots install `plotly`. For database connections install `DBI` plus `RSQLite` or `RPostgres`. For GIS maps install `sf`.

Value

Launches the Shiny app (does not return).

Examples

```
## Not run:
launch_app()

## End(Not run)
```

---

load_geometries *Load excavation geometries from file or database*

Description

Reads stratigraphic unit polygons from Shapefile, GeoPackage, GeoJSON, or a PostGIS database for use with gg_map.

Usage

```
load_geometries(
  source,
  layer = NULL,
  query = NULL,
  us_column = "us",
  crs = NULL
)
```

Arguments

source	File path or DBIConnection.
layer	Layer name for multi-layer sources.
query	SQL query for database connections.
us_column	Column containing US/context identifiers.
crs	Target CRS for reprojection (optional).

Value

An sf object with a us column.

See Also

gg_map, read_db

Other data-import: read_db(), read_pyarchinit()

---

local_sei *Compute local SEI values*

Description

Aggregates the SEI matrix by row, yielding a per-observation measure of total depositional coherence with all other finds.

Usage

```
local_sei(sei_mat)
```

Arguments

sei_mat	A symmetric SEI matrix from sei_matrix.

Value

A numeric vector of length nrow(sei_mat).

See Also

sei_matrix

Other SEI: sei_matrix(), sei_sparse()

Examples

```
x <- archaeo_sim(n = 30, k = 2, seed = 1)
S <- sei_matrix(x)
lsei <- local_sei(S)
summary(lsei)
```

`optimize_weights` *Estimate optimal SEI weights via cross-validation*

Description

Tests a grid of weight configurations and selects the one that maximises the mean held-out log-likelihood across folds. This provides a data-driven alternative to manual weight specification.

Usage

```
optimize_weights(
  data,
  k,
  weight_grid = NULL,
  n_folds = 3,
  seed = 1,
  verbose = TRUE,
  var_structure = "spherical",
  ...
)
```

Arguments

`data`	A data.frame with archaeological find data.
`k`	Number of phases.
`weight_grid`	A data.frame with columns `ws`, `wz`, `wt`, `wc`. If `NULL`, a default grid is used.
`n_folds`	Number of cross-validation folds (default: 3).
`seed`	Random seed.
`verbose`	Print progress (default: TRUE).
`var_structure`	Variance structure used for the weight-selection fits (default: `"spherical"`; see Details).
`...`	Additional arguments passed to `fit_sef`.

Details

Since v0.14.0 the weights scale the feature dimensions and therefore enter the mixture likelihood (in earlier versions they only affected the SEI matrix, so the cross-validated objective was invariant to them). Two technical points make the comparison meaningful. First, free diagonal variances absorb any rescaling of the feature columns, so weight selection is performed under `var_structure = "spherical"`, where the weights define the relative precision of the dimensions and are identifiable; the returned `best_weights` are therefore meant to be used with `fit_sef(..., var_structure = "spherical")`. Second, held-out log-likelihoods are mapped back to the common unweighted feature scale via a Jacobian correction ($n_{test} \sum_d \log w_d$) so that configurations with different weights are directly comparable.

Value

A list with components:

best_weights Named numeric vector of optimal weights.

results Data.frame with all tested configurations and their scores.

See Also

fit_sef, cv_sef

Other validation: adjusted_rand_index(), bootstrap_sef(), confusion_matrix(), cv_sef()

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
opt <- optimize_weights(x, k = 3, n_folds = 3)
opt$best_weights
```

---

pdi *Compute Palimpsest Dissolution Index*

Description

Measures global phase separability as $1 - \bar{H}/\log(K)$, where $\bar{H}$ is the mean per-find Shannon entropy of the soft phase assignments and K the number of phases. Values close to 1 indicate well-separated phases (low entropy); values near 0 indicate a compressed, heavily mixed palimpsest. As a rule of thumb, PDI > 0.7 indicates well-resolved phases. The index is an interpretable, dimensionless summary reported alongside the BIC (see compare_k) during model selection; unlike the BIC it does not penalise model complexity, so it should not be used on its own to choose K.

Usage

```
pdi(object)
```

Arguments

object	A sef_fit object.

Value

A single numeric value between 0 and 1.

See Also

fit_sef, compare_k

Other diagnostics: as_phase_table(), detect_intrusions(), ese(), phase_composition(), phase_diagnostic_table(), predict_phase(), recommend_setup(), type_longevity()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
pdi(fit)
```

`phase_composition` *Per-phase class composition*

Description

Returns the estimated categorical class profile of each phase from a fit made with `class_model = "multinomial"`: row `j` gives the probability that a find assigned to phase `j` belongs to each material class. This is the model-based counterpart of cross-tabulating finds by phase and class, and is directly interpretable (e.g. "phase 2 is dominated by amphorae and fine ware").

Usage

```
phase_composition(object)
```

Arguments

`object`	A `sef_fit` object fitted with `class_model = "multinomial"`.

Value

A data.frame with a `phase` column and one column per class level; each row sums to 1.

See Also

`fit_sef`, `as_phase_table`

Other diagnostics: `as_phase_table()`, `detect_intrusions()`, `ese()`, `pdi()`, `phase_diagnostic_table()`, `predict_phase()`, `recommend_setup()`, `type_longevity()`

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
fit <- fit_sef(x, k = 3)
phase_composition(fit)
```

phase_diagnostic_table

Return a compact diagnostic table

Description

Combines the input data with dominant phase, phase probabilities, entropy, local SEI, and energy in a single data.frame.

Usage

```
phase_diagnostic_table(object)
```

Arguments

object	A sef_fit object.

Value

A data.frame with all input columns plus diagnostics.

See Also

as_phase_table, as_sf_phase

Other diagnostics: as_phase_table(), detect_intrusions(), ese(), pdi(), phase_composition(), predict_phase(), recommend_setup(), type_longevity()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
pdt <- phase_diagnostic_table(fit)
names(pdt)
```

phase_transition_matrix

Phase vertical transition matrix

Description

Computes how often finds from phase i are found directly above finds from phase j in the vertical sequence, revealing the stratigraphic ordering of phases.

Usage

```
phase_transition_matrix(object)
```

Arguments

object	A sef_fit object.

Value

A $K \times K$ matrix where entry [i,j] counts transitions from phase i (above) to phase j (below).

See Also

us_summary_table

Other export: export_results(), us_summary_table()

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
fit <- fit_sef(x, k = 3)
phase_transition_matrix(fit)
```

---

plot_energy *Plot local Excavation Stratigraphic Energy (base R)*

Description

Plot local Excavation Stratigraphic Energy (base R)

Usage

```
plot_energy(object)
```

Arguments

object	A sef_fit object.

Value

Invisibly returns the object.

See Also

gg_energy for the ggplot2 version.

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_entropy(), plot_phasefield(), plot_sei_profile()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
plot_energy(fit)
```

`plot_entropy` *Plot entropy across space (base R)*

Description

Plot entropy across space (base R)

Usage

```
plot_entropy(object)
```

Arguments

`object`	A `sef_fit` object.

Value

Invisibly returns the object.

See Also

`gg_entropy` for the ggplot2/plotly version.

Other plotting: `as_plotly()`, `gg_bootstrap()`, `gg_compare_k()`, `gg_confusion()`, `gg_convergence()`, `gg_cv()`, `gg_direction()`, `gg_energy()`, `gg_entropy()`, `gg_intrusions()`, `gg_map()`, `gg_outliers()`, `gg_phase_composition()`, `gg_phase_profile()`, `gg_phasefield()`, `gg_unit_coherence()`, `gg_weights()`, `plot_energy()`, `plot_phasefield()`, `plot_sei_profile()`

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
plot_entropy(fit)
```

plot_phasefield *Plot dominant phase assignment (base R)*

Description

Plot dominant phase assignment (base R)

Usage

```
plot_phasefield(object)
```

Arguments

object	A sef_fit object.

Value

Invisibly returns the object.

See Also

gg_phasefield for the ggplot2/plotly version.

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_sei_profile()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
plot_phasefield(fit)
```

plot_sei_profile *Plot ordered SEI profile (base R)*

Description

Plot ordered SEI profile (base R)

Usage

```
plot_sei_profile(object)
```

Arguments

object	A sef_fit object.

Value

Invisibly returns the object.

See Also

sei_matrix, local_sei

Other plotting: as_plotly(), gg_bootstrap(), gg_compare_k(), gg_confusion(), gg_convergence(), gg_cv(), gg_direction(), gg_energy(), gg_entropy(), gg_intrusions(), gg_map(), gg_outliers(), gg_phase_composition(), gg_phase_profile(), gg_phasefield(), gg_unit_coherence(), gg_weights(), plot_energy(), plot_entropy(), plot_phasefield()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
plot_sei_profile(fit)
```

---

predict_phase *Predict phase probabilities*

Description

Convenience alias for as_phase_table.

Usage

```
predict_phase(object)
```

Arguments

object	A sef_fit object.

Value

A data.frame with probabilities and diagnostics.

See Also

as_phase_table

Other diagnostics: as_phase_table(), detect_intrusions(), ese(), pdi(), phase_composition(), phase_diagnostic_table(), recommend_setup(), type_longevity()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
head(predict_phase(fit))
```

read_db *Read archaeological data from a SQL database*

Description

Loads find data from any DBI-compatible database and maps columns to the format expected by `fit_sef`.

Usage

```
read_db(
  conn,
  query = NULL,
  table = NULL,
  col_id = "id",
  col_x = "x",
  col_y = "y",
  col_z = "z",
  col_context = "context",
  col_date_min = "date_min",
  col_date_max = "date_max",
  col_class = "class",
  col_taf = NULL,
  schema = "custom",
  sito = NULL
)
```

Arguments

`conn`	A `DBIConnection` object.
`query`	SQL query returning data with the required columns.
`table`	Table name (alternative to `query`).
`col_id, col_x, col_y, col_z, col_context, col_date_min, col_date_max, col_class, col_taf`	Column name mappings.
`schema`	Use `"pyarchinit"` for automatic PyArchInit mapping.
`sito`	Site filter for PyArchInit schema.

Value

A data.frame ready for `fit_sef`.

See Also

`fit_sef`, `load_geometries`

Other data-import: `load_geometries()`, `read_pyarchinit()`

read_harris *Read Harris Matrix from CSV edge list*

Description

Reads a CSV file with columns `from`, `to`, and optionally `weight`, and converts it to an $n \times n$ penalty matrix aligned with the find-level data.

Usage

```
read_harris(file, contexts, default_weight = 1)
```

Arguments

`file`	Path to CSV with columns `from`, `to`, and optionally `weight`.
`contexts`	Character vector of context labels for each find (length = number of finds).
`default_weight`	Weight for edges without an explicit weight.

Value

An $n \times n$ penalty matrix.

See Also

`harris_from_contexts`, `fit_sef`

Other harris: `harris_from_contexts()`, `validate_phases_harris()`

read_pyarchinit *Read finds from a pyArchInit database for palimpsest analysis*

Description

Reads finds from a pyArchInit database and assembles a data.frame ready for `fit_sef`. Finds come from `inventario_materiali_table` (materials) and/or `pottery_table` (pottery), selectable via `source`. A find uses its own plan coordinates when it is plotted as a point in `pyarchinit_reperti`; otherwise it inherits its stratigraphic unit's coordinates (centroid of the US polygon). Chronology comes from `us_table`, while elevation follows archaeological practice (see *Details*).

Usage

```
read_pyarchinit(
  con,
  us_geometry = NULL,
  sito = NULL,
  source = c("both", "materials", "pottery"),
  date_labels = NULL,
  taf = NULL,
  us_geom_field = NULL,
  synthetic_coords = TRUE,
  quote_table = "pyarchinit_quote",
  chronology_table = "palimpsest_chronology",
  pottery_class = c("ware", "material", "form"),
  reperti_table = "pyarchinit_reperti",
  reperti_geometry = NULL
)
```

Arguments

con	A DBIConnection to a pyArchInit database (SQLite/Spatialite or PostgreSQL/PostGIS).
us_geometry	Optional sf object of US polygons (e.g. read from the pyunitastratigrafiche layer); its centroids give each find's x/y.
sito	Optional site filter (matched against the sito column).
source	One of "both" (default), "materials", or "pottery" - which finds tables to read.
date_labels	Optional named list of period label -> c(min, max) year bounds, extending or overriding the built-in dictionary.
taf	Optional taphonomic score: a single value applied to all finds, or a vector named by find id. When NULL (default) and the chronology_table carries a per-US taf column, those per-US values are used; units without a value default to 0.5 (neutral).
us_geom_field	Name of the US-identifier column in us_geometry (auto-detected among us_s/us when NULL).
synthetic_coords	Logical (default TRUE). When a stratigraphic unit has no polygon in us_geometry, assign it deterministic per-unit grid coordinates so the analysis can still run at unit resolution; set FALSE to drop such finds instead.
quote_table	Name of the US-elevation point table (default "pyarchinit_quote"); set NULL to skip it.
chronology_table	Name of an optional per-US absolute-chronology table (default "palimpsest_chronology") with columns sito, area, us, start, end (calendar years, BCE negative) - e.g. OxCal-calibrated ranges from chronology_from_oxcal. When present it overrides the free-text datazione dating for the matching units (envelope of multiple dates per US); set NULL to skip it. An optional taf column in this table supplies the per-US taphonomic score (used when taf is NULL).

`pottery_class` Character vector of `pottery_table` columns; the find's `class` is the first non-empty of these per row (default `c("ware", "material", "form")`).

`reperti_table` Name of the pyArchInit piece-plotted-finds point layer (default `"pyarchinit_reperti"`); when a material find is plotted there (matched by site + `numero_inventario` = `id_rep`), its own point `x`, `y` (and `z`) replace the US-centroid coordinates. Set `NULL` to skip it. Falls back to the US centroid for finds without a point.

`reperti_geometry` Optional pre-read `sf` POINT layer of the finds (with `id_rep`/`siti`); when `NULL` (default) the layer is read from `con` (via `reperti_table`) if present.

Details

Elevation (`z`) is sourced, in order of precedence:

1. the find's own `quota_usm` (materials only);
2. the mean US elevation from `pyarchinit_quote` (`quota_q`), joined on `(sito, area, us)` with an `(sito, us)` fallback;
3. `0` when neither is available.

The deprecated `us_table.quota_abs` form field is **not** used. Pottery has no elevation column, so it always inherits its US elevation. Dating is resolved from `us_table.datazione` (falling back to the find's own dating) by an internal archaeological-date parser.

Value

A data.frame with `id`, `x`, `y`, `z`, `date_min`, `date_max`, `class`, `context`, `taf_score` (plus `find_source` when `source = "both"`), ready for `fit_sef`.

See Also

`read_db`, `fit_sef`

Other data-import: `load_geometries()`, `read_db()`

---

recommend_setup *Inspect a dataset and recommend* `fit_sef()` *flags*

Description

Examines the structural properties of a palimpsestr-ready dataset and returns a set of recommended `fit_sef()` options based on simple heuristics. Useful when the dataset has subtle properties (US-centroid coordinates, US-tied chronology, imbalanced classes, informative taphonomic scores) that would otherwise be discovered only by trial and error.

Usage

```
recommend_setup(
  data,
  coords = c("x", "y", "z"),
  chrono = c("date_min", "date_max"),
  class_col = "class",
  tafonomy = NULL,
  context = NULL
)
```

Arguments

data	A data.frame with archaeological find data.
coords	Character vector of length 3 naming the x, y, z coordinate columns (default: c("x", "y", "z")).
chrono	Character vector of length 2 naming the min and max chronology columns (default: c("date_min", "date_max")).
class_col	Character scalar with the class column name (default: "class").
tafonomy	Optional column name for taphonomic scores.
context	Optional column name for stratigraphic unit labels.

Value

An S3 object of class sef_setup with:

- recommendations: named list with logical flags class_scale, taf_as_feature, chrono_precision, residuality.
- diagnostics: list of structural properties detected in the data (number of classes, singleton classes, whether coordinates are at unit centroid, whether chronology is US-tied, whether taf is informative, etc.).
- warnings: character vector of caveats (e.g. directional intrusion will be uninformative because chronology is US-tied).
- suggested_call: string with the recommended fit_sef() invocation.

See Also

fit_sef

Other diagnostics: as_phase_table(), detect_intrusions(), ese(), pdi(), phase_composition(), phase_diagnostic_table(), predict_phase(), type_longevity()

Examples

```
x <- archaeo_sim(n = 100, k = 3, seed = 1)
recommend_setup(x, context = "context")


data(villa_romana)
```

```
recommend_setup(villa_romana, context = "context", tafonomy = "taf_score")
```

`reorder_phases` *Reorder phases by mean depth*

Description

Relabels phases so that phase 1 corresponds to the deepest (oldest) stratum and phase K to the shallowest (most recent). This ensures consistent, interpretable phase numbering across different runs.

Usage

```
reorder_phases(object)
```

Arguments

`object`	A `sef_fit` object.

Value

A `sef_fit` object with reordered phases.

See Also

`fit_sef`

Other fitting: `compare_k()`, `fit_sef()`, `sef_summary()`

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
fit <- fit_sef(x, k = 3)
fit <- reorder_phases(fit)
```

report_sef *Generate a textual interpretive report for a SEF model*

Description

Produces a structured Markdown report covering phase composition, intrusion detection, stratigraphic unit purity, and recommendations. Available in English and Italian.

Usage

```
report_sef(object, lang = c("en", "it"), file = NULL)
```

Arguments

object	A sef_fit object.
lang	Language: "en" (English) or "it" (Italian).
file	Optional file path to save the report.

Value

Character string with the report text (invisibly).

See Also

fit_sef, sef_summary

Examples

```
x <- archaeo_sim(n = 100, k = 3, seed = 1)
fit <- fit_sef(x, k = 3)
report_sef(fit, lang = "en")
```

sef_summary *Compact summary for a fitted SEF model*

Description

Returns a named list with global diagnostics and phase counts.

Usage

```
sef_summary(object)
```

Arguments

object	A sef_fit object.

Value

A named list.

See Also

fit_sef, pdi

Other fitting: compare_k(), fit_sef(), reorder_phases()

Examples

```
x <- archaeo_sim(n = 60, k = 2, seed = 1)
fit <- fit_sef(x, k = 2)
sef_summary(fit)
```

---

sei_matrix *Compute the Stratigraphic Entanglement Index matrix*

Description

Builds an $n \times n$ symmetric matrix quantifying pairwise depositional coherence from spatial, vertical, temporal, and cultural evidence.

Usage

```
sei_matrix(
  data,
  coords = c("x", "y", "z"),
  chrono = c("date_min", "date_max"),
  class_col = "class",
  weights = c(ws = 1, wz = 1, wt = 1, wc = 1),
  eps = 1e-09,
  z_floor = 0.25,
  max_dist = NULL
)
```

Arguments

data	Input data.frame.
coords	Character vector of coordinate column names (x, y, z).
chrono	Character vector with minimum and maximum dating columns.
class_col	Class column name.
weights	Named numeric vector with components ws, wz, wt, wc. Each component is normalised to $[0, 1]$ before weighting, so the weights represent relative importance.
eps	Deprecated and ignored (kept for backward compatibility). Since v0.14.0 the spatial component uses a bounded exponential kernel with a data-driven bandwidth, so no epsilon guard is needed.

z_floor	Deprecated and ignored (kept for backward compatibility). Since v0.14.0 the vertical component uses a bounded exponential kernel with a data-driven bandwidth, so no floor is needed.
max_dist	Maximum spatial distance for pair inclusion. When NULL (default), all pairs are computed. For large datasets (n > 2000), setting this to a reasonable neighbourhood radius dramatically reduces memory and computation time. The result is a sparse-like matrix with zeros for distant pairs.

Value

A symmetric numeric matrix with zero diagonal.

Note

SEI values are normalised within each dataset. Absolute SEI values are **not directly comparable** across different excavations or datasets of different sizes. Use SEI for within-dataset ranking and relative comparisons only.

See Also

local_sei, fit_sef

Other SEI: local_sei(), sei_sparse()

Examples

```
x <- archaeo_sim(n = 30, k = 2, seed = 1)
S <- sei_matrix(x)
dim(S)
```

---

sei_sparse *Compute SEI matrix with automatic sparsification*

Description

Wrapper around sei_matrix that automatically sets max_dist when the dataset exceeds a size threshold, using the 25th percentile of pairwise distances as the cutoff.

Usage

```
sei_sparse(
  data,
  coords = c("x", "y", "z"),
  chrono = c("date_min", "date_max"),
  class_col = "class",
  weights = c(ws = 1, wz = 1, wt = 1, wc = 1),
  eps = 1e-09,
  z_floor = 0.25,
  n_threshold = 1000
)
```

Arguments

data	Input data.frame.
coords	Character vector of coordinate column names (x, y, z).
chrono	Character vector with minimum and maximum dating columns.
class_col	Class column name.
weights	Named numeric vector with components ws, wz, wt, wc. Each component is normalised to $[0, 1]$ before weighting, so the weights represent relative importance.
eps	Deprecated and ignored (kept for backward compatibility). Since v0.14.0 the spatial component uses a bounded exponential kernel with a data-driven bandwidth, so no epsilon guard is needed.
z_floor	Deprecated and ignored (kept for backward compatibility). Since v0.14.0 the vertical component uses a bounded exponential kernel with a data-driven bandwidth, so no floor is needed.
n_threshold	Datasets larger than this trigger sparse mode (default: 1000).

Value

A numeric matrix (same as sei_matrix).

See Also

sei_matrix

Other SEI: local_sei(), sei_matrix()

Examples

```
x <- archaeo_sim(n = 50, k = 2, seed = 1)
S <- sei_sparse(x)
```

---

summary.sef_fit *Summarise a fitted SEF model*

Description

Summarise a fitted SEF model

Usage

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

Arguments

object	A sef_fit object.
...	Ignored.

Value

A named list.

`type_longevity` *Posterior-weighted type longevity*

Description

For each cultural class (or sub-class), returns the temporal envelope of the depositional phases in which the class has non-negligible posterior membership. The longevity is the union of the per-phase chronological envelopes for those phases whose mean posterior weight (averaged over the finds of that class) exceeds `posterior_threshold`.

Usage

```
type_longevity(
  object,
  class_col = NULL,
  posterior_threshold = 0.1,
  sub_class = FALSE
)
```

Arguments

`object`	A `sef_fit` object produced by `fit_sef()`.
`class_col`	Optional. Column name to use as the class label. Defaults to the `class` argument used in `fit_sef()`.
`posterior_threshold`	Numeric in `[0, 1]`. Phases with mean posterior weight below this threshold are excluded from the envelope (default: 0.1).
`sub_class`	Logical. If TRUE and `object$subclass` is set, the class label is the cross factor `class:subclass` (default: FALSE).

Value

A data.frame with columns `class`, `longevity_min`, `longevity_max`, `longevity_span`, `dominant_phase`, `n_finds`, and a list-column `weight_matrix` containing the per-phase posterior weights (length K).

See Also

`detect_intrusions`, `gg_longevity`

Other diagnostics: `as_phase_table()`, `detect_intrusions()`, `ese()`, `pdi()`, `phase_composition()`, `phase_diagnostic_table()`, `predict_phase()`, `recommend_setup()`

Examples

```
x <- archaeo_sim(n = 90, k = 3, seed = 1)
fit <- fit_sef(x, k = 3, context = "context")
type_longevity(fit)
```

`us_summary_table` *Summary table per stratigraphic unit*

Description

Aggregates finds by context (US), reporting the dominant phase, purity (proportion of finds in dominant phase), mean entropy, mean energy, and intrusion count.

Usage

```
us_summary_table(object)
```

Arguments

`object`	A `sef_fit` object.

Value

A data.frame with one row per stratigraphic unit.

See Also

`export_results`, `phase_diagnostic_table`

Other export: `export_results()`, `phase_transition_matrix()`

Examples

```
x <- archaeo_sim(n = 80, k = 3, seed = 1)
fit <- fit_sef(x, k = 3, context = "context")
us_summary_table(fit)
```

validate_phases_harris *Validate phase assignments against stratigraphic ordering*

Description

Checks whether the estimated phases follow the expected vertical ordering within each stratigraphic unit pair.

Usage

```
validate_phases_harris(object)
```

Arguments

object	A sef_fit object.

Value

A data.frame with one row per context pair, indicating whether the dominant phase ordering is consistent with depth.

See Also

harris_from_contexts, us_summary_table

Other harris: harris_from_contexts(), read_harris()

Examples

```
x <- archaeo_sim(n = 60, k = 3, seed = 1)
fit <- fit_sef(x, k = 3, context = "context")
validate_phases_harris(fit)
```

villa_romana *Poggio Gramignano Roman Villa excavation dataset*

Description

Real archaeological dataset from the site of Poggio Gramignano (Lugnano in Teverina, TR, Italy), a multi-period Roman villa with an annexed Late Antique infant cemetery. The dataset contains 615 inventoried finds from 54 stratigraphic units spanning from the Pre-Roman period (7th–4th c. BCE) through the Late Antique (5th–6th c. CE).

Usage

```
villa_romana
```

Format

A data.frame with 615 rows and 9 variables:

id Unique find identifier (VRPG_0001 to VRPG_0615)

x Easting coordinate (metres, EPSG:3004)

y Northing coordinate (metres, EPSG:3004)

z Mean elevation of the stratigraphic unit (metres a.s.l.)

context Stratigraphic unit label (e.g. US_76, US_117)

date_min Start of chronological interval (BCE as negative, CE as positive)

date_max End of chronological interval

class Material class (e.g. Reperto Ceramico, Reperto Anforaceo)

taf_score Taphonomic integrity score (0 = fully disturbed, 1 = pristine in situ)

Details

Chronological data derives from the site periodisation. Coordinates are US centroids in Monte Mario / Italy zone 2 (EPSG:3004). Elevation (z) is the mean of recorded quotes per US. Taphonomic scores were assigned by the excavator based on depositional context (primary, secondary, redeposition, backfill, disturbance).

Source

Data from the pyArchInit database of Poggio Gramignano (VRPG 3004). Taphonomic scores compiled by Roberto Montagnetti.

Examples

```
data(villa_romana)
str(villa_romana)
table(villa_romana$class)
```

Index