Package 'yaap'

June 7, 2026

Title A Toolkit for Archetypal Analysis Methods

Version 1.0.0

Description Fits archetypal analysis models, including Euclidean, probabilistic, kernel, and directional variants. Methods include classical archetypal analysis from Cutler and Breiman (1994) <doi:10.1080/00401706.1994.10485840>, PCHA and kernel variants from Mørup and Hansen (2012) <doi:10.1016/j.neucom.2011.06.033>, probabilistic archetypal analysis from Seth and Eugster (2016) <doi:10.1007/s10994-015-5498-8>, directional archetypal analysis from Olsen et al. (2022) <doi:10.3389/fnins.2022.911034>, AA++ initialization from Mair and Sjölund (2023) <doi:10.48550/arXiv.2301.13748>, coreset-style initialization from Mair and Brefeld (2019) <https://proceedings.neurips.cc/paper_files/paper/2019/file/7f278ad602c7f47aa76d1bfc90f20263-Paper.pdf>, and adapted AIC from Suleman (2017) <doi:10.1109/FUZZ-IEEE.2017.8015385>. Provides initialization helpers, model selection paths, plotting methods, 'broom' methods, and a 'tidymodels' recipe step.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports generics (>= 0.1.3), graphics, methods, Matrix, matrixStats, nnls, rlang (>= 1.0.0), stats, tibble (>= 3.0.0), utils, vctrs

Suggests archetypes, bench, compositions, fda, geometry, ggplot2, ggtern, irlba, knitr, MASS, quadprog, recipes (>= 1.0.0), rmarkdown, RSpectra, testthat (>= 3.0.0), tune (>= 1.0.0), withr (>= 2.5.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

VignetteBuilder knitr

NeedsCompilation no

Author Teo Sakel [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-9946-9498>),
MCIU/AEI [fnd] (ROR: <https://ror.org/05r0vyz12>, DOI:
10.13039/501100011033)

Maintainer Teo Sakel <teo@intelligentbiodata.com>

Repository CRAN

Date/Publication 2026-06-07 18:40:02 UTC

Contents

aa_init *Archetypal Analysis Initialization Functions*

Description

Various initialization methods for Archetypal Analysis (see Details).

Usage

```
aa_init(
  X,
  K,
  method = "furthest_sum",
  sparse = inherits(X, "sparseMatrix"),
  batch_size = NULL,
  batch_type = c("distal", "uniform"),
  batch_replace = FALSE,
  hull_method = c("full", "projected", "partitioned"),
  projected_dim = 2L,
  n_partitions = 10L,
  n_projection_max = NULL,
  use_unique_candidates = FALSE,
  ...
)
```

Arguments

X	a numeric matrix (rows = samples, columns = dimensions)
K	number of archetypes to be initialized
method	initialization method. One of "random", "dirichlet", "furthest_first", "kmeans_pp", "furthest_sum", "aa_pp", or "hull_outmost" (default: "furthest_sum").
sparse	whether B should be a sparse matrix (default: same as X)
batch_size	optional number of candidate rows to sample.
batch_type	candidate sampling strategy, "distal" or "uniform".
batch_replace	whether to sample candidate batches with replacement.
hull_method	strategy used by "hull_outmost". One of "full", "projected", or "partitioned" (default: "full").
projected_dim	projection dimension used by "hull_outmost" when hull_method = "projected" (default: 2L).
n_partitions	number of row partitions used by "hull_outmost" when hull_method = "partitioned" (default: 10L).
n_projection_max	optional maximum number of random projections used by "hull_outmost" for hull_method = "projected". If NULL, all feature projections are evaluated.

use_unique_candidates
: whether "hull_outmost" should de-duplicate hull candidates before vote tallying (default: FALSE).

...
: additional arguments (used for compatibility with user-defined functions)

Details

aa_init() runs the selected initialization method and formats the result as a named list containing the archetype coordinates A and row-stochastic matrix B. User-defined initialization functions passed directly to archetypes_nnls() or archetypes_pgd() must follow that same interface.

The following initialization methods are available:

- "random": selects uniformly at random K rows of X as archetypes.
- "dirichlet": draws each archetype as a random convex combination of all data points by sampling B row-wise from a Dirichlet(alpha, ..., alpha) distribution (Gamma(alpha, 1) weights normalized to unit sum). alpha = 1 (default) gives a uniform distribution over all data points; alpha < 1 concentrates mass near K data points, yielding sparser rows; alpha > 1 concentrates mass toward the centroid. Accepts optional alpha argument via ....
- "furthest_first": selects the first archetype randomly and then greedily selects the point furthest from the current set of archetypes.
- "kmeans_pp": a soft version of furthest_first where points are sampled in proportion to their distance from the current set of archetypes instead of greedily picking the furthest point every time.
- "furthest_sum": selects the first archetype randomly and then greedily selects the next archetypes that maximizes the sum of distances of all points from the current set of archetypes (see Mørup & Hansen 2012).
- "aa_pp": AA++ is a probabilistic initialization method similar to kmeans_pp but instead using the distances to the current set of archetypes it uses distances to their convex hull (see Mair & Sjölund 2023).
- "hull_outmost": computes hull candidates using one of the hull_method strategies ("full", "projected", or "partitioned") and then selects K archetypes via an outmost-vote ranking. This family of hull-based initializations is adapted from the **archetypal** package (Mouselimis et al., 2025).

Supplying batch_size applies the selected method to a candidate batch rather than to all rows. For all methods except "aa_pp", one batch is sampled up front. For "aa_pp", a fresh batch is sampled at each approximation step; this is a variant of the Monte Carlo AA++ approximation rather than the exact "aa_pp_mc" scheme. batch_replace = FALSE by default.

The default batch_type = "distal" is the coreset sampling strategy of Mair & Brefeld (2019): rows farther from the data center are more likely to be candidates, which saves memory and time while focusing on the boundary where archetypes are expected to lie. Use batch_type = "uniform" for an unbiased candidate batch.

Value

a named list containing two matrices: the archetype coordinates A and a row-stochastic matrix B such that A = B %*% X.

References

Mørup, M., & Hansen, L. K. (2012). Archetypal analysis for machine learning and data mining. Neurocomputing, 80, 54-63. doi:10.1016/j.neucom.2011.06.033

Mair, S., & Sjölund, J. (2023). Archetypal analysis++: Rethinking the initialization strategy. https://arxiv.org/abs/2301.13748

Mair, S., & Brefeld, U. (2019). Coresets for archetypal analysis. Advances in Neural Information Processing Systems, 32. https://proceedings.neurips.cc/paper_files/paper/2019/file/7f278ad602c7f47aa76d1bfc90f20263-Paper.pdf

Mouselimis, L., et al. (2025). **archetypal**: Archetypal Analysis with Principal Convex Hull Analysis. R package version 1.3.1. https://cran.r-project.org/package=archetypal

---

`AIC.archetypes` *AIC for archetypes objects*

Description

Computes the AIC-like validity criterion for an `archetypes` object.

Usage

```
## S3 method for class 'archetypes'
AIC(object, ...)
```

Arguments

`object`	An object of class `archetypes`.
`...`	Ignored.

Details

This is not the classical likelihood-based Akaike Information Criterion $-2\log L + 2k$. Instead, it implements the adapted archetypal-analysis criterion proposed by Suleman (2017), using the reconstruction error and an efficiency-adjusted complexity penalty. The value is intended for comparing Euclidean Gaussian archetype fits on the same data, especially across different numbers of archetypes `K`. For `K = 1`, the efficiency adjustment is undefined and `AIC()` returns `NA_real_`.

Value

Numeric scalar with the adapted AIC-like criterion.

References

A. Suleman, "Validation of archetypal analysis" 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, Italy, 2017, pp. 1-6, doi:10.1109/FUZZIEEE.2017.8015385

anames *Archetype names*

Description

Get or set the names of archetypes in an archetype analysis result.

Usage

```
anames(x)

anames(x) <- value

## S3 method for class 'archetypes'
anames(x)

## S3 replacement method for class 'archetypes'
anames(x) <- value
```

Arguments

x	An archetype analysis result.
value	Character vector with one name per archetype.

Value

anames() returns a character vector. The replacement method returns x with names updated consistently across archetype coordinates, coefficients, compositions, and initial coordinates when present.

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
fit <- run_aa(as.matrix(toy), K = 3, max_iter = 20, tol_r2 = 0.95)
anames(fit)
anames(fit) <- c("A", "B", "C")
anames(fit)
```

`archetypes_directional` *Directional Archetypal Analysis*

Description

Fits directional archetypal analysis, where data and archetypes are defined as directions (axes passing through the origin) rather than points in Euclidean space.

Usage

```
archetypes_directional(
  x,
  K,
  init = "dirichlet",
  init_args = list(),
  weights = NULL,
  max_iter = 100L,
  tol = 1e-04,
  tol_r2 = 0.9999,
  eps = 1e-08,
  verbose = FALSE,
  hemisphere = c("pca", "none"),
  precision = c("row_norm", "unit"),
  step_size = 1,
  max_iter_optimizer = 10L,
  step_shrinkage = 0.5,
  max_no_update = 5L
)
```

Arguments

`x`	dense numeric data matrix (rows = samples, columns = dimensions).
`K`	number of archetypes.
`init`	initialization method; see `run_aa()` for available options.
`init_args`	list of additional arguments for the initialization function.
`weights`	optional non-negative sample weights (default: NULL)
`max_iter`	maximum number of outer iterations (default: 100)
`tol`	convergence tolerance based on directional residual loss (default: 1e-4)
`tol_r2`	convergence tolerance based on directional R^2 (default: 0.9999)
`eps`	small positive number for numerical stability (default: 1e-8)
`verbose`	whether to print progress messages (default: FALSE)
`hemisphere`	hemisphere handling. `"pca"` flips generator rows onto the dominant PCA hemisphere; `"none"` leaves signs unchanged (default: `"pca"`)

`precision`	precision weighting. `"row_norm"` keeps row magnitudes in the Watson loss, matching Olsen et al.; `"unit"` gives every row equal precision (default: `"row_norm"`)
`step_size`	initial projected-gradient step size (default: 1.0)
`max_iter_optimizer`	maximum line-search iterations per update (default: 10)
`step_shrinkage`	factor used to shrink rejected line-search steps (default: 0.5)
`max_no_update`	maximum consecutive outer iterations with no accepted line-search update before stopping as stalled (default: 5)

Details

Directional AA:

Standard AA compares points by Euclidean distance, so long vectors can dominate short ones even when they point in the same direction. Directional AA instead compares observations on the sphere using a Watson loss: the spherical analogue of a Gaussian loss when opposite points are treated as the same direction. The fitted model minimizes the corresponding negative log-likelihood proxy, so a sample and its antipodal reflection contribute equally.

Hemisphere alignment:

The Watson loss treats a unit vector and its negation as identical. Without alignment, opposing signs can produce phantom "average direction" archetypes near the equator.

`hemisphere = "pca"` (default) projects each row onto the first principal component and flips rows with a negative dot product, so all generators lie on the same side. `hemisphere = "none"` leaves signs unchanged; use this when rows are already sign-aligned, when signs should be preserved, or when alignment has been handled upstream. With unaligned data, antipodal rows may cancel in the convex archetype construction even though the Watson loss treats them as equivalent.

Precision weighting:

The Watson loss weights each sample's angular residual by the squared norm of its reconstruction. `precision = "row_norm"` (default) retains the original row magnitudes, giving naturally larger observations higher influence matching the formulation in Olsen et al. (2022). `precision = "unit"` normalises every row to unit length before computing the loss, treating all samples as equally reliable.

Initialization:

The default `dirichlet` initialization draws the `coefficient` matrix as a Dirichlet(1, ..., 1), which is the simplex equivalent of the uniform distribution. This differs from `random` (the `aa_init()` method that selects K rows of the data uniformly at random), which initializes `B` as a one-hot matrix so that archetypes start at actual data points on the hemisphere.

Value

An object of class `directional_archetypes`, extending `archetypes`. Directional methods define unit-normalized fitted values and residuals, composition prediction for new directional data, and report `AIC()` as unsupported for Watson-loss fits.

References

Olsen, A. S., Høegh, R. M. T., Hinrich, J. L., Madsen, K. H., & Mørup, M. (2022). Combining electro- and magnetoencephalography data using directional archetypal analysis. *Frontiers in Neuroscience*, 16, 911034. doi:10.3389/fnins.2022.911034

See Also

`run_aa()` for the common entry point and full parameter documentation.

Examples

```
theta <- seq(0, pi / 2, length.out = 50)
X <- cbind(cos(theta), sin(theta))
fit <- archetypes_directional(X, K = 3, max_iter = 5)
```

---

archetypes_kernel_pgd *Kernel Archetypal Analysis using Projected Gradient Descent*

Description

Fits archetypal analysis from pairwise kernel similarities.

Usage

```
archetypes_kernel_pgd(
  x,
  K,
  kernel = c("rbf", "laplace", "polynomial", "linear", "precomputed"),
  kernel_args = list(),
  data = NULL,
  init = "furthest_sum",
  init_args = list(),
  robust = FALSE,
  robust_args = list(),
  max_iter = 100L,
  tol = 1e-04,
  tol_r2 = 0.9999,
  eps = 1e-08,
  verbose = FALSE,
  delta = 0,
  pseudo_pgd = TRUE,
  step_size = 1,
  max_iter_optimizer = 10L,
  step_shrinkage = 0.5,
  max_no_update = 5L
)
```

Arguments

x	data matrix (rows = samples) or an N x N Gram matrix when kernel = "precomputed".
K	number of archetypes.
kernel	kernel specification. One of "linear", "rbf", "laplace", "polynomial", "precomputed", or a function taking two data matrices and returning their cross-kernel matrix.
kernel_args	list of arguments passed to the kernel.
data	optional original data matrix attached to precomputed-kernel fits so coordinates() can return an input-space proxy.
init	kernel initialization. Accepts a method name string, a function, or a numeric K x N coefficient matrix whose rows are non-negative and sum to the allowed archetype mass. Matrix-valued init is interpreted as generator weights over samples, not as K x M input-space coordinates.
init_args	list of additional arguments for method-string or function initialization.
robust	robust row reweighting selector. See run_aa() for details. (default: FALSE)
robust_args	list of tuning arguments passed to the robust psi function.
max_iter	maximum number of outer iterations (default: 100)
tol	convergence tolerance based on residual sum of squares (default: 1e-4)
tol_r2	convergence tolerance based on R^2 (default: 0.9999)
eps	small positive number to ensure numerical stability (default: 1e-8)
verbose	whether to print progress messages (default: FALSE)
delta	maximum allowed relaxation of archetype convexity constraint (default: 0)
pseudo_pgd	whether to use pseudo projected gradient descent (default: TRUE)
step_size	initial line-search step size (default: 1.0)
max_iter_optimizer	maximum line-search iterations per update (default: 10)
step_shrinkage	factor used to shrink rejected line-search steps (default: 0.5)
max_no_update	maximum consecutive outer iterations with no accepted line-search update before stopping as stalled (default: 5)

Details

Kernels and the Gram matrix:

A kernel function $k(x_i, x_j)$ measures pairwise similarity between samples in some transformed feature space. The $N \times N$ Gram matrix with entries $G_{ij} = k(x_i, x_j)$ is the only input the solver needs. Built-in kernels and their kernel_args:

linear tcrossprod(X).

rbf Gaussian RBF: exp(-0.5 * dist(X)^2 / sigma^2).

laplace Laplace: exp(-dist(X, method = "manhattan") / sigma).

polynomial (gamma * tcrossprod(X) + coef0)^degree.

Kernel parameters passed via kernel_args:

- `sigma`: bandwidth for `rbf` and `laplace`; auto-selected when `NULL`.
- `gamma`: scale for `polynomial` (default `1/ncol(x)`).
- `degree`: polynomial degree (default 3).
- `coef0`: polynomial intercept (default 1).

Note: a "linear" kernel is equivalent to standard pgd AA on the original data but less efficient.

To use a precomputed Gram matrix, compute it externally, pass it as `x`, and set `kernel = "precomputed"`. Supply the original data as the `data` argument if you want coordinates() to return an input-space proxy.

Initialization:

Kernel archetypes are optimized as generator weights over the `N` training samples. Consequently, a matrix supplied to `init` must be a `K x N` coefficient matrix `B`, where row `k` defines the initial archetype $\sum_i B_{ki}\phi(x_i)$ in feature space. To initialize from specific training samples, construct this coefficient matrix explicitly with `onehot()`, for example `onehot(c(1, 5, 9), sparse = FALSE, nc = N)`. Numeric, character, and logical row selectors are not accepted directly as kernel `init` values.

The supported method strings are `"random"`, `"furthest_first"`, `"kmeans_pp"`, `"furthest_sum"`, and `"dirichlet"`. Candidate batching can be requested through `init_args = list(batch_size = ..., batch_type = ...)`; distal batching uses distances implied by the Gram matrix. A custom initializer function is called with `G`, `K`, and `init_args`, and must return either a `K x N` coefficient matrix or a list with component `B`. Unlike Euclidean fitters, kernel fits do not accept a `K x M` matrix of input-space archetype coordinates as `init`.

Archetype coordinates:

Because the space where similarities are measured is defined by the kernel, archetype coordinates are not directly available as ordinary input-space coordinates. The coordinates() accessor returns a proxy when the original input data are stored: A = B X, using the fitted generator weights B. For linear kernels this equals the usual Euclidean representation. For nonlinear kernels it is only a plotting and inspection proxy; distances in this coordinate matrix do not carry kernel-metric meaning.

When the original data are unavailable, as with some precomputed Gram matrices, coordinates() returns NULL. For an alternative feature-space view, plot.kernel_archetypes() with projection = "pca" projects archetype compositions onto kernel PCA components of the data.

Value

An object of class `kernel_archetypes`, extending `archetypes`. Kernel methods expose Hilbert-space residual norms, input-space coordinate proxies when data are stored, and out-of-sample `predict(type = "compositions")` via cross-kernel evaluation.

References

Mørup, M., & Hansen, L. K. (2012). Archetypal analysis for machine learning and data mining. *Neurocomputing*, 80, 54-63. doi:10.1016/j.neucom.2011.06.033

See Also

`run_aa()` for the common entry point and full parameter documentation.

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
archetypes_kernel_pgd(as.matrix(toy), K = 3)
```

`archetypes_nnls` *Archetypal Analysis using Non-Negative Least Squares*

Description

Performs archetypal analysis by iteratively solving the linear systems definining each of the 3 components (`coordinates`, `compositions`, `coefficients`) using non-negative least-squares (NNLS) to enforce the simplex constraints.

Usage

```
archetypes_nnls(
  x,
  K,
  init = "furthest_sum",
  init_args = list(),
  weights = NULL,
  scale = FALSE,
  robust = FALSE,
  robust_args = list(),
  sd_threshold = 1e-06,
  max_iter = 100L,
  tol = 1e-04,
  tol_r2 = 0.9999,
  max_kappa = 1000,
  eps = ifelse(inherits(x, "sparseMatrix"), 0, 1e-08),
  verbose = FALSE,
  max_no_update = 5L,
  bigM = NULL
)
```

Arguments

`x`	data matrix (rows = samples, columns = dimensions)
`K`	number of archetypes
`init`	initialization method; see `run_aa()` for available options.
`init_args`	list of additional arguments for the initialization function.
`weights`	optional vector of sample weights (default: NULL)
`scale`	scaling or metric embedding used before fitting; see `archetypes_pgd()` for details (default: FALSE).

robust	robust row reweighting selector. See run_aa() for available options (default: FALSE).
robust_args	list of tuning arguments passed to the robust psi function.
sd_threshold	threshold for feature standard deviation to filter low-variance features (default: 1e-6).
max_iter	maximum number of iterations (default: 100)
tol	convergence tolerance based on residual sum of squares (default: 1e-4)
tol_r2	convergence tolerance based on R^2 (default: 0.9999)
max_kappa	maximum condition-number warning threshold for NNLS diagnostics (default: 1000). Use Inf to disable condition-number warnings.
eps	small positive number to ensure numerical stability (default: 0 for sparse input 1e-8 for dense)
verbose	whether to print progress messages (default: FALSE)
max_no_update	maximum consecutive iterations without improvement before considering NNLS stalled (default: 5)
bigM	large constant to enforce simplex constraint, or NULL to set it automatically.

Details

NNLS solver:

Standard AA formulation involves 3 linear systems $X = SA = S(BX)$, where S is the compositions matrix, A is the coordinates matrix and B is the coefficients matrix. archetypes_nnls() solves each of these sequentially, via least-squares minimization, keeping the other two matrices fixed. For the matrices, S and B, that are subject to simplex constraints (non-negativity and row-sum-to-one), the NNLS solver is used and the sum-to-one constraint is enforced by adding a penalty term bigM into the systems involving these matrices.

When bigM = NULL (default) its value is set automatically based on heuristics. If it fails to enforce the simplex constraint, warning messages will ask you to increase it. If the solver is slow or numerically unstable try decreasing it.

Because the solution involves solving linear systems, the condition numbers of the compositions and coordinates matrices are tracked as diagnostics (k_S and k_A respectively). If either of them exceeds max_kappa, a warning is issued.

The returned loss history stores loss (residual sum-of-squares) as the best objective seen so far and rloss as the current iterate objective. The rloss, k_S, and k_A columns are NNLS diagnostics and are not used for model selection.

Value

An object of class archetypes.

References

Alcacer, A., Epifanio, I., Mair, S., & Mørup, M. (2025). A Survey on Archetypal Analysis. *arXiv preprint arXiv:2504.12392*. https://arxiv.org/abs/2504.12392

See Also

run_aa() for the common entry point and full parameter documentation.

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
archetypes_nnls(as.matrix(toy), K = 3)
```

---

archetypes_paa *Probabilistic Archetypal Analysis using Projected Gradient Descent*

Description

Fits probabilistic archetypal analysis (PAA) by minimising the log-likelihood of the data of several exponential-family model via projected gradient descent (PGD).

Usage

```
archetypes_paa(
  x,
  K,
  family = c("gaussian", "binomial", "poisson", "multinomial"),
  init = "furthest_sum",
  init_args = list(),
  max_iter = 100L,
  tol = 1e-04,
  tol_r2 = 0.9999,
  eps = 1e-08,
  verbose = FALSE,
  step_size = 1,
  max_iter_optimizer = 10L,
  step_shrinkage = 0.5,
  max_no_update = 5L
)
```

Arguments

x	data matrix (rows = samples, columns = dimensions).
K	number of archetypes.
family	observation family. One of "gaussian", "binomial", "poisson", or "multinomial" (default: "gaussian").
init	initialization method; see run_aa() for available options.
init_args	list of additional arguments for the initialization function.
max_iter	maximum number of outer iterations (default: 100)

`tol`	convergence tolerance based on objective loss (default: 1e-4)
`tol_r2`	convergence tolerance based on R^2 (default: 0.9999)
`eps`	small positive number for numerical stability (default: 1e-8)
`verbose`	whether to print progress messages (default: FALSE)
`step_size`	initial line-search step size (default: 1.0)
`max_iter_optimizer`	maximum line-search iterations per update (default: 10)
`step_shrinkage`	factor used to shrink rejected line-search steps (default: 0.5)
`max_no_update`	maximum consecutive outer iterations with no accepted line-search update before stopping as stalled (default: 5)

Details

Observation families:

PAA models each observation as drawn from an exponential-family distribution whose natural parameter is a convex combination of K archetypal profiles. Before optimisation begins, each data point is mapped to a fixed profile by computing its per-sample MLE parameter under the chosen family (e.g. the raw values for `"gaussian"`, high probability for `"binomial"`, etc.). The archetypes are then found as convex combinations of these fixed profiles. Built-in families and their data requirements:

`gaussian` Squared reconstruction error; data can be any real matrix.

`binomial` Binary cross-entropy; each entry must lie in [0, 1].

`poisson` Poisson log-likelihood; all entries must be non-negative.

`multinomial` Multinomial log-likelihood; entries must be non-negative with positive row sums, treated as count vectors.

Note: `gaussian` PAA is equivalent to standard AA; use `archetypes_pgd()` instead for a more efficient solver with support for missing data, sample weights, and metric scaling.

Value

An object of class `archetypes`.

References

Seth, S., & Eugster, M. J. A. (2016). Probabilistic archetypal analysis. *Machine Learning*, 102, 85-113. doi:10.1007/s1099401554988

See Also

`run_aa()` for the common entry point and full parameter documentation.

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
archetypes_paa(as.matrix(toy), K = 3)
```

archetypes_path *Fit an Archetypes Path Across K*

Description

Fits a sequence of archetypal analysis models over candidate numbers of archetypes. The input data are checked and preprocessed once, then the selected solver's fit step is run independently for each candidate K.

Usage

```
archetypes_path(x, K, ...)

## S3 method for class 'formula'
archetypes_path(x, K, data = NULL, ..., subset, na.action)

## Default S3 method:
archetypes_path(
  x,
  K,
  method = c("pgd", "nnls", "kernel", "directional", "paa"),
  family = "gaussian",
  init = NULL,
  init_args = list(),
  weights = NULL,
  scale = FALSE,
  robust = FALSE,
  robust_args = list(),
  sd_threshold = 1e-06,
  max_iter = 100L,
  tol = 1e-04,
  tol_r2 = 0.9999,
  eps = NULL,
  verbose = FALSE,
  missing = NULL,
  nrep = 1L,
  ...
)
```

Arguments

x	numeric matrix, data frame, formula, or supported specialized input. For formula input, the response, when present, is ignored.
K	candidate numbers of archetypes. A single value expands to 1:K; a vector is sorted and deduplicated.
...	additional arguments passed to run_aa().

`data` optional data frame supplying variables for formula input.

`subset` optional expression selecting rows before fitting formula input.

`na.action` function controlling missing-value handling for formula input. Defaults to `stats::na.omit()`.

`method` fitting method. One of `"pgd"`, `"nnls"`, `"kernel"`, `"directional"`, or `"paa"` (default: `"pgd"`).

`family` observation family passed to `method = "paa"`. Defaults to `"gaussian"`.

`init` initialization method for archetype starting coordinates. Accepts a function, a method name string, or a numeric initialization matrix. For `method = "kernel"`, see Details and `archetypes_kernel_pgd()`. `NULL` selects `"furthest_sum"` for all methods except `"directional"`, which defaults to `"dirichlet"`. Matrix row names are used as archetype names. Available method strings:

- `"furthest_sum"` greedily maximises the sum of distances from the current archetype set (Mørup & Hansen 2012). Default for most methods.
- `"furthest_first"` greedy farthest-point selection.
- `"kmeans_pp"` probabilistic farthest-point selection (soft furthest-first).
- `"random"` uniformly random sample of K rows.
- `"dirichlet"` random convex combinations sampled from a Dirichlet distribution. Default for `"directional"`.
- `"aa_pp"` AA++ initialization (Mair & Sjölund 2023). Pass `batch_size` in `init_args` to use a Monte Carlo-inspired variant.
- `"hull_outmost"` hull-candidate outmost-vote ranking.

`init_args` list of additional arguments for the initialization function. Any initializer can receive `batch_size`, `batch_type`, and `batch_replace` through `init_args`; `batch_type = "distal"` implements coreset-style candidate sampling (Mair & Brefeld 2019). See `vignette("initialization", package = "yaap")` for a comparison.

`weights` optional numeric vector of sample weights (default: `NULL`). Internally scaled to mean 1 and square-rooted before use.

`scale` common `run_aa()` scaling argument, present for consistency across method dispatch. Only Euclidean Gaussian methods (`"pgd"` and `"nnls"`) use it: `FALSE` (default) leaves columns on their original scale, `TRUE` applies z-score standardization, a positive numeric vector divides by user-supplied scale factors, and a symmetric positive-definite matrix applies the corresponding feature metric. Specialized methods (`"kernel"`, `"directional"`, and `"paa"`) define their own geometry or likelihood; non-`FALSE` values are ignored with a warning.

`robust` robust row reweighting selector. Use `FALSE` for ordinary squared error, a MASS psi function name (see `MASS::rlm()` for details), or a custom psi function. `TRUE` selects to `"psi.bisquare"`.

`robust_args` list of tuning arguments passed to the robust psi function.

`sd_threshold` threshold for feature standard deviation below which columns are dropped before fitting (default: 1e-6).

`max_iter` maximum number of outer iterations (default: 100).

`tol` convergence tolerance on the residual sum of squares (default: 1e-4).

tol_r2	convergence tolerance on R^2 (default: 0.9999).
eps	small positive number for numerical stability (default: 0 for sparse input, 1e-8 for dense).
verbose	whether to print progress messages (default: FALSE).
missing	logical or NULL; whether to fit the missing-data PGD objective. NULL auto-detects missing values in dense inputs. Only method = "pgd" supports TRUE; other methods set or require FALSE.
nrep	number of random restarts; the best fit (lowest final loss) is returned (default: 1).

Value

An object of class archetypes_path containing one archetypes fit per candidate K. Use [[with a numeric index or model name such as "K3" to extract one fitted model with the shared data restored. Use [to subset the path while preserving the archetypes_path container. Use $ for metadata such as K, method, family, and data.

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
path <- archetypes_path(as.matrix(toy), K = 3, max_iter = 20)
screeplot(path)
path[["K2"]]
```

---

archetypes_pgd *Archetypes Analysis using Projected Gradient Descent*

Description

Fits an archetypal analysis (AA) model by minimising the reconstruction error $\|X - SBX\|_F^2$, where S is the composition matrix and $A = BX$ is the archetype coordinates via projected gradient descent (PGD).

Usage

```
archetypes_pgd(
  x,
  K,
  init = "furthest_sum",
  init_args = list(),
  weights = NULL,
  scale = FALSE,
  robust = FALSE,
  robust_args = list(),
  sd_threshold = 1e-06,
  max_iter = 100L,
  tol = 1e-04,
```

```
  tol_r2 = 0.9999,
  eps = ifelse(inherits(x, "sparseMatrix"), 0, 1e-08),
  verbose = FALSE,
  missing = any(is.na(x)),
  delta = 0,
  pseudo_pgd = TRUE,
  step_size = 1,
  max_iter_optimizer = 10L,
  step_shrinkage = 0.5,
  max_no_update = 5L
)
```

Arguments

x	data matrix (rows = samples, columns = dimensions)
K	number of archetypes
init	initialization method; see run_aa() for available options.
init_args	list of additional arguments for the initialization function.
weights	optional vector of sample weights (default: NULL)
scale	scaling or metric embedding used before fitting. FALSE (default) leaves columns on their original scale, TRUE applies z-score preprocessing, a positive numeric vector divides columns by user-supplied scale factors, and a symmetric positive-definite matrix applies the corresponding feature metric embedding in the original data column space.
robust	robust row reweighting selector. Use FALSE for ordinary squared error, TRUE for "psi.bisquare", a MASS psi function name, or a custom psi function. See MASS::rlm() for psi details; method = "MM" is not supported because it is not applicable to AA.
robust_args	list of tuning arguments passed to the robust psi function.
sd_threshold	threshold for feature standard deviation to filter low-variance features (default: 1e-6)
max_iter	maximum number of iterations (default: 100)
tol	convergence tolerance based on residual sum of squares (default: 1e-4)
tol_r2	convergence tolerance based on R^2 (default: 0.9999)
eps	small positive number to ensure numerical stability (default: 0 for sparse input 1e-8 for dense)
verbose	whether to print progress messages (default: FALSE)
missing	whether to fit the missing-data PGD objective. When TRUE, only observed entries are optimized; dense NA values are treated as missing and sparse structural zeros are treated as missing.
delta	maximum allowed relaxation of archetypes convexity constraint (default: 0)
pseudo_pgd	whether to use pseudo projected gradient descent (default: TRUE)
step_size	initial step size for the gradient descent (default: 1.0)

`max_iter_optimizer`
: maximum iterations for the line search optimizer (default: 10)

`step_shrinkage`
: factor to reduce step size if no improvement during line search (default: 0.5)

`max_no_update`
: maximum consecutive outer iterations with no accepted line-search update before PGD stops as stalled (default: 5)

Details

Variants of the Gaussian AA formulation:

Three arguments change *which* version of AA is perfomed by altering the objective of relaxing the constraints.

- `delta` relaxes the convexity constraint on the archetypes. With the default `delta = 0`, every archetype is a convex combination of observed data points (i.e., it lies inside or on the boundary of the data convex hull). Setting `delta > 0` allows archetypes to step outside the hull by up to a factor $1 + \delta$ and inside by up to $1 - \delta$ — useful when the true extremes are absent from the data due to truncation or censoring. Use with care: large values remove the interpretability guarantee that archetypes are representable as extreme prototypes, and the uniqueness guarantees no longer hold.
- `robust` switches the loss from ordinary squared error to an iteratively re-weighted version based on MASS-style psi row weights. Use `TRUE` for `"psi.bisquare"`, or pass `"psi.huber"`, `"psi.hampel"`, or a custom psi function with tuning values in `robust_args`. See `MASS::rlm()` for psi details. `method = "MM"` from `MASS::rlm()` is not supported because MM estimation is not directly applicable to the AA objective. Robust fitting is incompatible with `missing = TRUE`.
- `missing` limits loss contributions to only observed entries and the archetype profiles are normalised entry-wise so that each archetype is a weighted average of the *observed* values of the data points that define it. For dense matrices, `NA` marks missing entries; for sparse `sparseMatrix` inputs, structural zeros are treated as missing. The default is `any(is.na(x))`, so missing-data mode is activated automatically when the input contains `NA`s.

For non-Gaussian variants of AA, see `vignette("non-gaussian-aa", package = "yaap")`

Algorithm mechanics and tuning:

The solver alternates gradient steps for S and B with an inner line search of up to `max_iter_optimizer` steps (default 10) that shrinks the step size by `step_shrinkage` (default 0.5) until the objective decreases. The outer loop repeats for at most `max_iter` iterations (default 100).

When `pseudo_pgd = TRUE` (the default), the gradients are projected onto the tangent space of the ℓ_1 unit sphere before the simplex projection step so as to allow larger steps without violating the constraints. This is the "pseudo-PGD" variant of Mørup & Hansen (2012). `pseudo_pgd = FALSE` uses use the plain gradient of the objective function with respect to each variable. In either case, after each step the variables are projected back onto the simplex.

`step_size` sets the initial step size for both S and B updates (and for α when `delta > 0`). If no step in the inner line search is accepted for `max_no_update` consecutive outer iterations (default 5), the solver declares the iterate stalled and stops early. Step sizes are also reduced after a failed outer iteration, so in practice the solver self-tunes step sizes down as it approaches a local minimum; there is usually no need to change `step_size` unless the default is far from the optimal scale for a particular dataset.

Convergence is declared when the relative decrease in RSS between successive accepted updates falls below `tol` (default 1e-4) *or* R^2 exceeds `tol_r2` (default 0.9999).

Preprocessing:
Before fitting, features with standard deviation below `sd_threshold` (default 1e-6) are dropped to avoid ill-conditioning; their values are restored as column means in the output.
The `scale` argument is best understood as choosing the feature metric used to judge reconstruction errors. It changes which directions in feature space are treated as close or far, so it can emphasize, de-emphasize, or correlate features without changing the sample-level convexity constraints. This is useful for curves, spectra, correlated measurements, or any setting where ordinary Euclidean distance is not the desired geometry. Use `weights`, not `scale`, when the goal is to make some samples count more than others. See `vignette("non-gaussian-aa", package = "yaap")` for examples.

Value

An object of class `archetypes`

References

Mørup, M., & Hansen, L. K. (2012). Archetypal analysis for machine learning and data mining. *Neurocomputing*, 80, 54-63. doi:10.1016/j.neucom.2011.06.033

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
archetypes_pgd(as.matrix(toy), K = 3)
```

---

augment.archetypes *Augment data with composition weights from an archetypes model*

Description

Adds per-sample archetype composition columns to the original data.

Usage

```
## S3 method for class 'archetypes'
augment(x, data = NULL, ...)

## S3 method for class 'kernel_archetypes'
augment(x, data = NULL, ...)
```

Arguments

`x`	An object of class `archetypes` or `kernel_archetypes`.
`data`	Optional data frame or matrix to augment. If `NULL`, uses the data stored inside `x` (if available). For `archetypes` objects, passing new data triggers `predict.archetypes()` to compute compositions.
`...`	Passed to `predict.archetypes()` when `data` is provided and `x` is an `archetypes` object.

Details

For `kernel_archetypes`, `data` is used only to convert the stored data to a tibble; compositions always come from the stored `compositions(x)` since projecting new samples requires the original Gram matrix.

Value

A tibble with all columns from `data` plus one column per archetype named `.A1`, `.A2`, etc. (dot-prefixed `anames(x)`).

See Also

`tidy.archetypes()`, `glance.archetypes()`

---

coefficients.archetypes

Coefficients for archetypes objects

Description

Returns the weights that express each archetype as a combination of samples.

Usage

```
## S3 method for class 'archetypes'
coefficients(object, ...)
```

Arguments

`object`	An object of class `archetypes`.
`...`	Ignored.

Value

A numeric matrix (K x N) where each row contains the sample weights used to form the corresponding archetype.

See Also

`fitted.archetypes()`, `predict.archetypes()`, `residuals.archetypes()`

Examples

```
fit <- run_aa(iris[, 1:4], K = 3)
coefficients(fit)
```

`consistency` *Consistency Between Archetypal Analysis Fits*

Description

Measures how similar two archetypal analysis fits are by comparing their memberships, archetype definitions, or archetype locations.

Usage

```
consistency(x, y, ...)

## S3 method for class 'archetypes'
consistency(
  x,
  y,
  what = c("compositions", "coefficients", "coordinates"),
  data = NULL,
  ...
)
```

Arguments

`x`	An object.
`y`	A fitted archetypes object to compare with `x`.
`...`	Arguments passed to methods.
`what`	Component to compare: compositions, coefficients, or coordinates.
`data`	Optional input data used as the variance reference for coordinate consistency. Defaults to `x$data`.

Details

Consistency is useful when checking whether an AA solution is stable across random starts, resampled data, or nearby model choices. A high score means the two fits describe the data in a similar way, even if archetype labels have changed. Compare `compositions` to ask whether samples are assigned to the same archetypal mixtures, `coefficients` to ask whether archetypes are built from the same samples, and `coordinates` to ask whether the archetype locations are close in the input space.

For `what = "compositions"` or `what = "coefficients"`, the two components are treated as row-stochastic membership matrices and compared with normalized mutual information (NMI). For matrices `X` and `Y`, the joint distribution is `Pxy = crossprod(X, Y) / nrow(X)`. Mutual information is computed from `Pxy` and its row and column marginals, then normalized as `2 * MI(X, Y) / (MI(X, X) + MI(Y, Y))`. This makes the score insensitive to permutations of the archetype labels.

For `what = "coordinates"`, archetypes in `x` are greedily matched to the nearest unmatched archetypes in `y`, then a R2-like score is computed by scaling the mean matched distances by the overall variation in `data`. This option is useful for checking whether two fits place archetypes in similar regions

of the input space. It returns NA when x has more archetypes than y, because not every archetype in x can be matched.

Value

A numeric scalar.

References

J. L. Hinrich, S. E. Bardenfleth, R. E. Røge, N. W. Churchill, K. H. Madsen, and M. Mørup, "Archetypal analysis for modeling multisubject fMRI data," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 7, pp. 1160-1171, 2016.

---

coordinates *Access archetype coordinates and compositions*

Description

coordinates() returns archetype coordinates in the same dimensions as the original input data. compositions() returns the weights that express each sample as a combination of archetypes.

Usage

```
coordinates(object, ...)

## S3 method for class 'archetypes'
coordinates(object, ...)

compositions(object, ...)

## S3 method for class 'archetypes'
compositions(object, ...)
```

Arguments

object	An archetype analysis result.
...	Ignored.

Details

For kernel fits coordinates() return an input-space proxy computed from the fitted generator weights and stored data, since the real coordinates live in the space where the kernel inner product is defined.

For non-Gaussian families , coordinates live in parameter space (logit for binomial, log for Poisson, etc.) and may not be directly comparable to the original data.

Value

`coordinates()` returns a numeric matrix with K rows of archetype coordinates. `compositions()` returns an N x K numeric matrix of composition weights.

`fitted.archetypes` *Fitted values for archetypes objects*

Description

Computes the projection of the data on the convex hull of the archetypes fit as `compositions %*% coordinates`.

Usage

```
## S3 method for class 'archetypes'
fitted(object, ...)
```

Arguments

`object`	An object of class `archetypes`.
`...`	Ignored.

Value

Fitted values in the input feature space for Gaussian fits and in the family parameter space for non-Gaussian fits.

See Also

`residuals.archetypes()`, `predict.archetypes()`, `coefficients.archetypes()`

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
fit <- run_aa(as.matrix(toy), K = 3, max_iter = 20, tol_r2 = 0.95)
Xhat <- fitted(fit)
dim(Xhat)
```

`fit_simplex` *Fit Data to Convex Hull defined by Archetypes*

Description

This function fits new data points to a convex hull of the given archetype coordinates by minimizing the square distance of the projection.

Usage

```
fit_simplex(
  A,
  X,
  method = c("nnls", "QP"),
  eps = 0,
  project = proj_l1,
  lambda = 1e-08,
  bigM = NULL
)
```

Arguments

`A`	Numeric matrix (K x M) of archetype coordinates.
`X`	Numeric matrix (N x M) of new data points to fit.
`method`	Character string specifying the fitting method: • `"nnls"`: Non-negative least squares with a soft sum-to-one constraint. • `"QP"`: Quadratic programming approach to directly fit onto simplex.
`eps`	Numeric scalar used for numerical stability; ensures non-negativity of fit.
`project`	Function used to project the `nnls` fit onto the simplex (default: `proj_l1`).
`lambda`	Numeric scalar used for numerical stability in `QP` solver (default: 1e-8).
`bigM`	Large constant used by `method = "nnls"` to softly enforce the simplex sum constraint. When `NULL`, selected by the same heuristic used by the NNLS fitter.

Details

The function solves the constrained least-squares problem that minimizes `norm(X - S %*% A, "F")` such that `all(S >= 0)` and `rowSums(S) = 1`.

The method `"nnls"` fits `S` via non-negative least squares using the `nnls` package after augmenting `A` and `X` with a large `bigM` column to softly enforce the sum-to-one constraint. If the raw NNLS row sums differ from one by more than 0.01, `bigM` is doubled up to three times. The result is then projected onto the simplex via the `project` method (by default `proj_l1`).

The method `"QP"` solves the full constrained least squares problem via quadratic programming using the `quadprog` package, enforcing both non-negativity and row-stochasticity constraints directly. This method may be more accurate but also slower for large data sets.

`lambda` acts as a L_2 regularization parameter to ensure numerical stability of the quadratic programming solver.

Value

Numeric row-stochastic matrix (N x K) of fitted compositions.

`glance.archetypes` *Glance at an archetypes model*

Description

Returns a one-row tibble of model-level summary statistics.

Usage

```
## S3 method for class 'archetypes'
glance(x, ...)

## S3 method for class 'kernel_archetypes'
glance(x, ...)
```

Arguments

`x`	An object of class `archetypes` or `kernel_archetypes`.
`...`	Ignored.

Value

A one-row tibble with columns:

`K` number of archetypes.

`converged` logical; did the optimizer converge?

`loss` final residual sum of squares.

`r2` final R^2.

`n_iter` number of iterations run (excluding the initialisation row).

`family` observation family (for `archetypes` fits only).

See Also

`tidy.archetypes()`, `augment.archetypes()`

onehot *One-hot encode a vector*

Description

onehot() converts a vector into a **one-hot encoded** matrix: each distinct value corresponds to a column, and each row has a 1 in the matching column and 0 elsewhere. This is useful for turning categorical values into numbers for modeling and data preprocessing.

Usage

```
onehot(ind, sparse = FALSE, ...)

## Default S3 method:
onehot(ind, sparse = FALSE, nc = NULL, ...)

## S3 method for class 'factor'
onehot(ind, sparse = FALSE, ...)

## S3 method for class 'character'
onehot(ind, sparse = FALSE, ...)
```

Arguments

ind	A vector to encode. Can be numeric/integer, factor, or character.
sparse	Logical; if TRUE, return a memory-efficient sparse matrix.
...	Additional arguments passed to the specific input handler.
nc	Number of columns (for numeric input only). If NULL, uses max(ind).

Details

- **Numeric/integer input:** values are treated as 1-based column indices. The number of columns (nc) defaults to the largest value in the input, but you can set nc manually if you need more columns (e.g., reserving space for unseen categories).
- **Factor input:** each factor level becomes a column; the number of columns is fixed to the number of levels and the columns are named by those levels.
- **Character input:** automatically converted to a factor before encoding.

Missing values produce rows of all zeros.

Value

A one-hot encoded matrix:

sparse = FALSE a dense matrix.

sparse = TRUE a sparse Matrix::dgCMatrix.

Methods (by class)

- onehot(default): Numeric indices
- onehot(factor): Factor method: nc is fixed to the number of levels
- onehot(character): Character method: delegates to factor

Examples

```
# Numeric vector: columns correspond to indices 1..max(ind)
onehot(c(1, 2, 1, 3))

# Factor: columns follow factor levels (and are named)
f <- factor(c("red", "blue", "red", NA, "green"), levels = c("red", "blue", "green"))
onehot(f)

# Character: automatically treated as a factor
onehot(c("cat", "dog", "cat"), sparse = TRUE)

# Reserving extra columns for future/unseen categories (numeric input only)
onehot(c(1, 2, 1), sparse = FALSE, nc = 5)
```

---

plot.archetypes *Plot method for archetypes objects*

Description

Draws diagnostic and geometric plots for a fitted archetypal analysis model.

Usage

```
## S3 method for class 'archetypes'
plot(
  x,
  what = c("compositions", "loss", "coordinates", "profiles"),
  subset = NULL,
  plot = TRUE,
  ...
)
```

Arguments

x	An object of class archetypes
what	Character string naming the plot to draw. Supported values are:
	"composition", "composision" Draw a stacked barplot of fitted samples compositions, with bars representing observations and colors representing archetypes. Rows and columns are clustered by default.

"ternary", "simplex" Compositions are plotted as points in the 2D simplex, with corners representing archetypes. For K > 3, multiple 2D projections are drawn. This plot requires package compositions.

"loss" Plot the evolution of the objective value across optimization iterations.

"profiles" Plot a barplot of the fitted archetypes. Each set of bars represents a dimension and archetypes are separated by color.

"coordinates" Scatterplot of archetype coordinates, optionally over the original observations when data are available.

subset	Optional sample subset for plots that display observations. May be numeric row indices, sample names, or a logical vector. Subsetting is applied before clustering or projecting.
plot	Logical. Should the plot be drawn? If FALSE, the prepared plotting data is returned without drawing.
...	Additional graphical parameters passed to the selected plotting helper.

Value

The prepared data used by the selected plot, returned invisibly when plot = TRUE.

---

plot_archetypes_compositions

Composition Plot For Archetypes

Description

Draws a horizontal stacked barplot for a matrix-like set of composition weights, with rows interpreted as samples and columns interpreted as archetypes or other compositional parts.

Usage

```
plot_archetypes_compositions(
  compositions,
  plot = TRUE,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  distance = NULL,
  distance_rows = "euclidean",
  distance_cols = "euclidean",
  linkage = "complete",
  col = NULL,
  legend = TRUE,
  border = NA,
  ...
)
```

Arguments

`compositions`	Numeric matrix or data frame. Rows are samples and columns are archetypes. Rows should contain non-negative composition weights.
`plot`	Logical. Should the plot be drawn?
`cluster_rows, cluster_cols`	Logical values, one of `"PC1"` or `"AOP"`, or `hclust` objects. When `TRUE`, rows or columns are reordered by hierarchical clustering. `"PC1"` orders rows by their first principal component score and columns by their first principal component loading. `"AOP"` orders rows by the angle of their PC1/PC2 scores and columns by the angle of their PC1/PC2 loadings.
`distance`	Optional distance metric used for both row and column clustering when `distance_rows` or `distance_cols` are not supplied.
`distance_rows, distance_cols`	Distance metrics used when clustering rows or columns. Values may be any method accepted by `stats::dist()`, `"correlation"`, a function that returns a `dist` object, or a precomputed `dist` object. Row distances are computed after the centered log-ratio transform.
`linkage`	Linkage method passed to `stats::hclust()`.
`col`	Optional vector of colors, one per archetype. Defaults to a qualitative HCL palette.
`legend`	Logical. Should an archetype legend be drawn?
`border`	Border color for the stacked bar segments.
`...`	Additional graphical parameters passed to `graphics::barplot()`.

Value

Returns a data frame in long format with one row per sample/archetype pair. Columns are `sample` (factor ordered by the applied clustering), `archetype` (factor ordered by the applied clustering), and `weight` (numeric).

`plot_archetypes_coordinates` *Coordinate Plot For Archetypes*

Description

Plots archetype coordinates, optionally overlaid on observation data. When `data` is supplied and `projection = "pca"`, observations and coordinates are projected together onto the first two principal components before plotting.

Usage

```
plot_archetypes_coordinates(
  coordinates,
  data = NULL,
  projection = c("none", "pca"),
  archetype_names = NULL,
  show_anames = TRUE,
  args.data.scatter = list(),
  plot = TRUE,
  ...
)
```

Arguments

coordinates	Numeric matrix of archetype coordinates. This is the only required argument.
data	Optional numeric matrix of observations to draw behind the archetypes.
projection	Projection to use. "pca" is only valid when data is supplied.
archetype_names	Optional labels to draw next to archetype points.
show_anames	Logical. Should archetype labels be drawn?
args.data.scatter	Named list of graphical arguments for observation points.
plot	Logical. Should the plot be drawn?
...	Graphical arguments for archetype points and paths. General plot window arguments such as main, xlab, ylab, xlim, ylim, and asp are also honored when drawing the plot.

Value

Returns a data frame in long format with one row per point. Columns are the coordinate dimensions, name (character label), and archetype (logical). Data rows come first, archetype rows last.

---

plot_archetypes_loss *Loss Plot For Archetypes*

Description

Loss Plot For Archetypes

Usage

```
plot_archetypes_loss(loss, plot = TRUE, ...)
```

Arguments

`loss`	Data frame containing a `loss` column.
`plot`	Logical. Should the plot be drawn?
`...`	Additional graphical parameters passed to `graphics::plot()`.

Value

Returns a data frame with columns `iteration` (integer, 0-based) and `loss` (numeric).

`plot_archetypes_profiles` *Profile Plot For Archetypes*

Description

Profile Plot For Archetypes

Usage

```
plot_archetypes_profiles(
  coordinates,
  family = "gaussian",
  archetype_names = NULL,
  plot = TRUE,
  ...
)
```

Arguments

`coordinates`	Archetype coordinates or an `fda::fd` object.
`family`	Observation family used to choose a default y-axis label.
`archetype_names`	Optional archetype labels.
`plot`	Logical. Should the plot be drawn?
`...`	Additional graphical parameters passed to `graphics::barplot()` for matrix coordinates, or `graphics::plot()` for `fd` coordinates.

Value

Returns a data frame in long format with columns `archetype`, `feature`, and `value`. Returns `NULL` invisibly for `fd` coordinates.

`predict.archetypes` *Predict compositions or reconstructions for new data from an archetypes model*

Description

Projects new samples onto the archetype space by solving for composition. The representation is either in the original coordinates (`type = "reconstruction"`) or as barycentric coordinates (`type = "compositions"`) on the archetype simplex.

Usage

```
## S3 method for class 'archetypes'
predict(object, newdata, type = c("reconstruction", "compositions"), ...)
```

Arguments

`object`	An object of class `archetypes`.
`newdata`	New data to fit. Must contain the features (columns) used to fit `object`; fd-backed fits may pass an `fda::fd` object.
`type`	Prediction output type. `"reconstruction"` (default) returns `X_hat = S %*% A`; `"compositions"` returns `S`.
`...`	Passed to `fit_simplex` and family-specific prediction routines.

Details

For non-Gaussian families, the reconstructions live in parameter space. In particular, "binomial" family returns probabilities and "poisson" family returns positive rates. For multinomial family, the reconstructions are category probabilities, so each row sums to one.

Value

If `type = "compositions"`, a numeric matrix (N_new x K) with non-negative row-stochastic composition weights. If `type = "reconstruction"`, a reconstruction matrix (N_new x M) in the input feature space.

See Also

`fit_simplex()`, `residuals.archetypes()`, `fitted.archetypes()`. For class-specific overrides see `predict.directional_archetypes()` and `predict.kernel_archetypes()`.

`predict.directional_archetypes` *Predict compositions or reconstructions for directional archetypes*

Description

Projects new directional samples onto the fitted archetype space, like `predict.archetypes()`. Directional reconstructions are returned as unit-length directions; `type = "compositions"` returns the archetype weights.

Usage

```
## S3 method for class 'directional_archetypes'
predict(
  object,
  newdata,
  type = c("reconstruction", "compositions"),
  max_iter = 100L,
  eps = 1e-08,
  step_size = 1,
  max_iter_optimizer = 10L,
  step_shrinkage = 0.5,
  ...
)
```

Arguments

`object`	An object of class `directional_archetypes`.
`newdata`	New directional data matrix.
`type`	Prediction output type. `"reconstruction"` (default) or `"compositions"`.
`max_iter`, `eps`, `step_size`, `max_iter_optimizer`, `step_shrinkage`	Optimization controls for composition fitting.
`...`	Ignored.

Value

A composition matrix (N_new x K) when `type = "compositions"`, or a directional reconstruction matrix (N_new x M) when `type = "reconstruction"`.

predict.kernel_archetypes *Predict method for kernel archetypes*

Description

Predict compositions for kernel archetypes

Usage

```
## S3 method for class 'kernel_archetypes'
predict(
  object,
  newdata,
  type = c("reconstruction", "compositions"),
  max_iter = 100L,
  eps = 1e-08,
  step_size = 1,
  max_iter_optimizer = 10L,
  step_shrinkage = 0.5,
  ...
)
```

Arguments

object	An object of class kernel_archetypes.
newdata	New data to project, or an N_new x N_train cross-kernel matrix when object was fit with kernel = "precomputed".
type	Prediction output type. Only "compositions" is defined.
max_iter, eps, step_size, max_iter_optimizer, step_shrinkage	Optimization controls for composition fitting.
...	Ignored.

Details

Projects new samples onto the fitted kernel archetype simplex. type = "compositions" returns the fitted simplex weights. type = "reconstruction" errors because inverse reconstruction from the implicit feature space is undefined for kernel archetypes.

Value

A numeric N_new x K row-stochastic composition matrix.

residuals.archetypes *Residuals for archetypes objects*

Description

Residuals for archetypes objects

Usage

```
## S3 method for class 'archetypes'
residuals(object, type = c("response", "pearson"), ...)
```

Arguments

object	An object of class archetypes.
type	Residual type. "response" (default) returns observed minus fitted values on the observation scale. "pearson" returns GLM-style Pearson residuals, dividing response residuals by the square root of the family variance function and applying stored row weights when present.
...	Ignored.

Value

Residuals in the same representation as data.

See Also

fitted.archetypes(), predict.archetypes(). For kernel fits see residuals.kernel_archetypes().

residuals.kernel_archetypes *Residuals for kernel archetypes objects*

Description

Residuals are per-sample squared distances in the implicitly defined feature space. The method is provided for completness and to support residual-based diagnostics as the natural fitted() method is not defined for nonlinear kernel archetypes.

Usage

```
## S3 method for class 'kernel_archetypes'
residuals(object, ...)
```

Arguments

`object`	An object of class `kernel_archetypes`.
`...`	Ignored.

Value

A named numeric vector of non-negative squared residual norms, one per sample.

See Also

`residuals.archetypes()` for Euclidean residuals on standard fits.

---

run_aa *Run Archetypal Analysis*

Description

Common entry point for fitting archetypal analysis models. Dispatches to the solver selected by `method`. For solver-specific arguments, theoretical background, and class-specific return slots, see the individual solver pages.

Usage

```
run_aa(x, ...)

## S3 method for class 'formula'
run_aa(formula, data = NULL, K, ..., subset, na.action)

## Default S3 method:
run_aa(
  x,
  K,
  method = c("pgd", "nnls", "kernel", "directional", "paa"),
  family = "gaussian",
  init = NULL,
  init_args = list(),
  weights = NULL,
  scale = FALSE,
  robust = FALSE,
  robust_args = list(),
  sd_threshold = 1e-06,
  max_iter = 100L,
  tol = 1e-04,
  tol_r2 = 0.9999,
  eps = NULL,
  verbose = FALSE,
```

```
  missing = NULL,
  nrep = 1L,
  ...
)

## S3 method for class 'fd'
run_aa(x, K, ...)
```

Arguments

x	numeric matrix (rows = samples, columns = features), or an object with a class-specific run_aa() method.
...	additional arguments passed to the selected solver. See archetypes_pgd(), archetypes_nnls(), archetypes_kernel_pgd(), archetypes_directional(), and archetypes_paa() for method-specific parameters.
formula	formula selecting variables from data. The response, when present, is ignored.
data	optional data frame supplying variables for formula input.
K	number of archetypes. When K has length greater than one, run_aa() returns an archetypes_path() object with one fit per value.
subset	optional expression selecting rows before fitting formula input.
na.action	function controlling missing-value handling for formula input. Defaults to stats::na.omit().
method	fitting method. One of "pgd", "nnls", "kernel", "directional", or "paa" (default: "pgd").
family	observation family passed to method = "paa". Defaults to "gaussian".
init	initialization method for archetype starting coordinates. Accepts a function, a method name string, or a numeric initialization matrix. For method = "kernel", see Details and archetypes_kernel_pgd(). NULL selects "furthest_sum" for all methods except "directional", which defaults to "dirichlet". Matrix row names are used as archetype names. Available method strings: "furthest_sum" greedily maximises the sum of distances from the current archetype set (Mørup & Hansen 2012). Default for most methods. "furthest_first" greedy farthest-point selection. "kmeans_pp" probabilistic farthest-point selection (soft furthest-first). "random" uniformly random sample of K rows. "dirichlet" random convex combinations sampled from a Dirichlet distribution. Default for "directional". "aa_pp" AA++ initialization (Mair & Sjölund 2023). Pass batch_size in init_args to use a Monte Carlo-inspired variant. "hull_outmost" hull-candidate outmost-vote ranking.
init_args	list of additional arguments for the initialization function. Any initializer can receive batch_size, batch_type, and batch_replace through init_args; batch_type = "distal" implements coreset-style candidate sampling (Mair & Brefeld 2019). See vignette("initialization", package = "yaap") for a comparison.

weights	optional numeric vector of sample weights (default: NULL). Internally scaled to mean 1 and square-rooted before use.
scale	common run_aa() scaling argument, present for consistency across method dispatch. Only Euclidean Gaussian methods ("pgd" and "nnls") use it: FALSE (default) leaves columns on their original scale, TRUE applies z-score standardization, a positive numeric vector divides by user-supplied scale factors, and a symmetric positive-definite matrix applies the corresponding feature metric. Specialized methods ("kernel", "directional", and "paa") define their own geometry or likelihood; non-FALSE values are ignored with a warning.
robust	robust row reweighting selector. Use FALSE for ordinary squared error, a MASS psi function name (see MASS::rlm() for details), or a custom psi function. TRUE selects to "psi.bisquare".
robust_args	list of tuning arguments passed to the robust psi function.
sd_threshold	threshold for feature standard deviation below which columns are dropped before fitting (default: 1e-6).
max_iter	maximum number of outer iterations (default: 100).
tol	convergence tolerance on the residual sum of squares (default: 1e-4).
tol_r2	convergence tolerance on R^2 (default: 0.9999).
eps	small positive number for numerical stability (default: 0 for sparse input, 1e-8 for dense).
verbose	whether to print progress messages (default: FALSE).
missing	logical or NULL; whether to fit the missing-data PGD objective. NULL auto-detects missing values in dense inputs. Only method = "pgd" supports TRUE; other methods set or require FALSE.
nrep	number of random restarts; the best fit (lowest final loss) is returned (default: 1).

Details

For method = "kernel", matrix-valued init is a K x N coefficient matrix over training samples, not a coordinate matrix. See archetypes_kernel_pgd() for the full kernel initialization contract.

Robust fitting is not supported for methods "directional" and "paa". Also compared to MASS::rlm() the "MM" mode is not supported in AA.

run_aa.fd() fits archetypal analysis to the coefficient matrix of an fda::fd object. The feature scaling is set to the basis inner-product matrix from fda::eval.penalty(data$basis, Lfdobj = 0), so scale cannot be supplied to this method. The internal optimization-space archetype coordinates are basis coefficients; coordinates() returns them in the original fd representation.

Value

An object of class archetypes with components:

coordinates (K x M) archetype coordinates in the original feature space.

coefficients (K x N) weights expressing each archetype as a convex combination of samples.

compositions (N x K) row-stochastic weights expressing each sample as a convex combination of archetypes.

`loss` data frame of per-iteration metrics. All fitters include `loss` and `r2`; additional diagnostic columns are method-specific.

`converged` logical convergence flag.

`data` original data passed to the fitter.

`call` the matched call.

`family` observation family string (e.g. `"gaussian"`).

`init` initial archetype coordinates (when available).

`slack`, `weights` optional relaxation and sample-weight parameters.

`feature_map` internal metadata mapping new data into the fitted optimization geometry for prediction.

See Also

Solvers: `archetypes_pgd()`, `archetypes_nnls()`, `archetypes_kernel_pgd()`, `archetypes_directional()`, `archetypes_paa()`. Post-fit: `plot.archetypes()`, `predict.archetypes()`, `fitted.archetypes()`, `residuals.archetypes()`, `anames()`. Model selection: `AIC.archetypes()`. Tidy output: `tidy.archetypes()`, `glance.archetypes()`, `augment.archetypes()`.

Examples

```
toy <- read.csv(system.file("extdata", "toy.csv", package = "yaap"))
run_aa(as.matrix(toy), K = 3)
run_aa(as.matrix(toy), K = 3, method = "nnls")
run_aa(Species ~ ., data = iris, K = 3)

# Functional data example based on fda::growth
if (requireNamespace("fda", quietly = TRUE)) {
    data(growth, package = "fda")
    basis_fd <- fda::create.bspline.basis(c(1, nrow(growth$hgtm)), 10)
    temp_fd <- fda::Data2fd(
        argvals = seq_len(nrow(growth$hgtm)),
        y = growth$hgtm,
        basisobj = basis_fd
    )

    fit_fd <- run_aa(temp_fd, K = 3, max_iter = 20, tol_r2 = 0.95)
    arch_fd <- coordinates(fit_fd)
}
```

`screeplot.archetypes_path` *Scree Plot for an Archetypes Path*

Description

Draws or prepares a model-selection curve over candidate `K` values.

Usage

```
## S3 method for class 'archetypes_path'
screeplot(x, y = NULL, plot = TRUE, ...)
```

Arguments

x	An archetypes_path object.
y	Metric to plot. NULL defaults to "AIC" for Euclidean Gaussian pgd and nnls paths, and to "r2" otherwise. A character value may be "AIC" or a column from each fit's loss table. A function is called on each extracted fit and must return a single numeric value.
plot	Logical. Should the plot be drawn?
...	Additional graphical parameters passed to graphics::plot().

Value

Invisibly returns a data frame with one row per candidate K.

---

simplex_projection *Project rows of matrix onto the probability simplex*

Description

These functions project each row of a numeric matrix onto the probability simplex or the L1-norm ball.

Usage

```
proj_simplex(mat, eps = 0)

proj_l1(mat, eps = 0)
```

Arguments

mat	A numeric matrix where each row will be projected.
eps	A small positive number to ensure numerical stability (default: 1e-8).

Details

The values of the input matrix are first clipped at eps to be non-negative and then projected onto the simplex or L1-norm ball.

The proj_simplex() function implements the efficient algorithm of Condat (2016). The proj_l1() function simply normalizes each row to sum to 1 after clipping.

Value

A row-stochastic matrix.

References

Condat, L. (2016). Fast projection onto the simplex and the L_1 ball. Mathematical Programming, 158(1), 575-585. doi:10.1007/s1010701509466

Examples

```
mat <- matrix(runif(12), nrow = 4)

proj_simplex(mat)

proj_l1(mat)
```

`step_archetypes` *Archetypal Analysis Preprocessing Step for recipes*

Description

`step_archetypes()` creates a *specification* of a recipe step that projects numeric predictors onto a K-archetype simplex. During `prep()` the archetypes are fitted once (with optional random-restart via `nrep`); during `bake()` new data are projected onto the learnt simplex.

Usage

```
step_archetypes(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  num_comp = 3L,
  delta = 0,
  fit_method = "pgd",
  options = list(),
  reconstruct = FALSE,
  keep_original_cols = FALSE,
  res = NULL,
  col_names = NULL,
  seed = sample.int(100000L, 1L),
  skip = FALSE,
  id = recipes::rand_id("archetypes")
)
```

Arguments

`recipe`	A recipe object.
`...`	One or more selector functions to choose which variables are affected by the step. See `recipes::selections()` for more details.

role	Role for the *new* columns produced by this step. Default "predictor".
trained	Logical. Has the step been trained (i.e. has prep() been called)? Set by prep() — do not change manually.
num_comp	Number of archetypes K (default 3L). Tunable via tune::tune().
delta	Convexity penalty for the pgd solver (default 0). Ignored when fit_method != "pgd". Tunable via tune::tune().
fit_method	Solver to use: "pgd" (default), "nnls", or "paa".
options	A named list of additional arguments passed to run_aa() (e.g. nrep, max_iter, scale). These are not validated here.
reconstruct	If TRUE, bake() also appends columns named rec_<original_col> containing the low-rank reconstruction S %*% Z (compositions times archetype coordinates).
keep_original_cols	Should the original predictor columns be retained after baking? Default FALSE.
res	The fitted archetypes object. NULL before prep(), set automatically by prep().
col_names	Character vector of selected column names. Set by prep().
seed	Integer seed for reproducible fitting. Evaluated once at construction time (default: a random integer).
skip	Should the step be skipped during bake() on the test set? Default FALSE.
id	A character string that identifies this step in the recipe.

Details

step_archetypes() is **not** supported for method = "directional" or method = "kernel".

The num_comp and delta parameters support tune::tune() for hyperparameter search.

Value

An updated recipe with the new step appended.

Examples

```
if (requireNamespace("recipes", quietly = TRUE)) {
    rec <- recipes::recipe(Species ~ ., data = iris) |>
        step_archetypes(
            recipes::all_numeric_predictors(),
            num_comp = 3L,
            options = list(max_iter = 20L, tol_r2 = 0.95)
        ) |>
        recipes::prep(training = iris)
    recipes::bake(rec, new_data = iris)
}
```

`tidy.archetypes` *Tidy an archetypes model*

Description

Converts an `archetypes` model into a tidy long-form tibble.

Usage

```
## S3 method for class 'archetypes'
tidy(x, matrix = c("coordinates", "coefficients", "compositions"), ...)

## S3 method for class 'kernel_archetypes'
tidy(x, matrix = c("coordinates", "coefficients", "compositions"), ...)
```

Arguments

`x`	An object of class `archetypes` or `kernel_archetypes`.
`matrix`	Which component to return: `"coordinates"` (K x M archetype coordinates, default), `"coefficients"` (K x N weights of archetypes over samples), or `"compositions"` (N x K weights of samples over archetypes).
`...`	Ignored.

Details

For `kernel_archetypes`, `matrix = "coordinates"` returns the `coordinates` matrix (`coefficients %*% data`) when available, and emits a warning with an empty tibble otherwise. `"coefficients"` and `"compositions"` behave identically to the `archetypes` method.

Value

A tibble. Column names depend on `matrix`:

`"coordinates"` `archetype`, `term`, `value` columns (K * M rows).

`"coefficients"` `archetype`, `sample`, `value` columns (K * N rows).

`"compositions"` `sample`, `archetype`, `value` columns (N * K rows).

See Also

`glance.archetypes()`, `augment.archetypes()`

Index