

# Package: geosmooth (via r-universe)

October 8, 2026

**Title** Geometric Methods for Nonparametric Regression and Density Estimation

**Version** 0.2.0

**Description** Provides geometric methods for nonparametric regression and density estimation on data represented as coordinate matrices or weighted graphs. Methods include local polynomial smoothing, model-averaged local polynomial smoothing, local polynomial lifting trend filtering, synchronized local polynomial lifting trend filtering, graph low-pass filtering, and Hessian-energy regression. Methodological references include Gajer and Ravel (2025) ``Adaptive Geometric Regression for High-Dimensional Structured Data" <doi:10.48550/arXiv.2511.03817>, Fan and Gijbels (1996, ISBN:9780412983214), Wang et al. (2016) ``Trend Filtering on Graphs" <<https://www.jmlr.org/papers/v17/15-147.html>>, and Kim et al. (2009) ``Semi-Supervised Regression Using Hessian Energy" <<https://papers.nips.cc/paper/3741-semi-supervised-regression-using-hessian-energy-with-an-application-to-semi-supervised-dimer>>

**Copyright** file inst/COPYRIGHTS

**License** GPL (>= 3)

**URL** <https://pgajer.github.io/geosmooth/>,  
<https://github.com/pgajer/geosmooth>

**BugReports** <https://github.com/pgajer/geosmooth/issues>

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**SystemRequirements** GNU make

**LinkingTo** Rcpp

**Depends** R (>= 3.5.0)

**Imports** dgraphs (>= 0.2.1.9000), digest, jsonlite, MASS, Matrix, methods, parallel, Rcpp, stats, utils

**Suggests** genlasso, grip ( $\geq 0.2.0.9001$ ), knitr, rmarkdown, testthat  
( $\geq 3.0.0$ ), waldo, withr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** cmake libglpk-dev make libxml2-dev

**Repository** <https://pgajer.r-universe.dev>

**Date/Publication** 2026-10-08 19:38:27 UTC

**RemoteUrl** <https://github.com/pgajer/geosmooth>

**RemoteRef** HEAD

**RemoteSha** 5e8959743d1634b2d7369dd213eb18ebce646a74

## Contents

|                                           |    |
|-------------------------------------------|----|
| geosmooth-package . . . . .               | 4  |
| apply.metric.graph.lowpass.path . . . . . | 5  |
| as.data.frame.synthetic_dataset . . . . . | 7  |
| bootstrap.malps . . . . .                 | 7  |
| compare.masked.graph.fields . . . . .     | 9  |
| compare.synthetic.dataset . . . . .       | 9  |
| cv.masked.graph.field . . . . .           | 10 |
| fit.chart.kernel . . . . .                | 11 |
| fit.density . . . . .                     | 13 |
| fit.graph.trend.filtering . . . . .       | 15 |
| fit.local.likelihood . . . . .            | 17 |
| fit.lpl.tf . . . . .                      | 20 |
| fit.lps . . . . .                         | 22 |
| fit.malps . . . . .                       | 28 |
| fit.masked.graph.field . . . . .          | 32 |
| fit.metric.graph.lowpass . . . . .        | 33 |
| fit.ps.lps . . . . .                      | 35 |
| fit.pttf.trend.filtering . . . . .        | 39 |
| fit.slpl.tf . . . . .                     | 42 |
| fit.ssrhe.hessian.l1.regression . . . . . | 44 |
| fit.ssrhe.hessian.regression . . . . .    | 49 |
| fit.subject.od . . . . .                  | 54 |
| get.region.boundary . . . . .             | 56 |
| graph.trend.filtering.operator . . . . .  | 58 |
| harmonic.smoother . . . . .               | 59 |
| lpl.tf.operator . . . . .                 | 62 |
| lps.backend.diagnostics . . . . .         | 64 |
| lps.grouped.foldid . . . . .              | 65 |
| lps.nested.cv . . . . .                   | 66 |
| lps.pointwise.band . . . . .              | 68 |
| malps.gcv . . . . .                       | 70 |

|                                                       |     |
|-------------------------------------------------------|-----|
| materialize.synthetic . . . . .                       | 71  |
| materialize.synthetic.instance . . . . .              | 72  |
| metric.graph.heat.eta.grid . . . . .                  | 73  |
| metric.graph.heat.extend.lower . . . . .              | 74  |
| metric.graph.lowpass.basis . . . . .                  | 75  |
| metric.graph.lowpass.operator . . . . .               | 77  |
| model-summary . . . . .                               | 79  |
| model-values . . . . .                                | 81  |
| normalize.density . . . . .                           | 84  |
| perform.harmonic.smoothing . . . . .                  | 86  |
| plot.harmonic_smoother . . . . .                      | 88  |
| plot.synthetic_dataset . . . . .                      | 89  |
| predict.lpl_tf . . . . .                              | 90  |
| predict.lps . . . . .                                 | 91  |
| predict.malps . . . . .                               | 91  |
| predict.slpl_tf . . . . .                             | 92  |
| print.graph.trend.filtering.fit . . . . .             | 93  |
| print.harmonic_smoother . . . . .                     | 95  |
| print.summary.harmonic_smoother . . . . .             | 96  |
| print.synthetic_dataset . . . . .                     | 97  |
| pttf.geometry . . . . .                               | 97  |
| pttf.operator . . . . .                               | 99  |
| pttf.operator.filter.rows . . . . .                   | 101 |
| refit . . . . .                                       | 102 |
| refit.lpl_tf . . . . .                                | 103 |
| refit.malps . . . . .                                 | 104 |
| refit.metric.graph.lowpass.fit . . . . .              | 106 |
| refit.slpl_tf . . . . .                               | 107 |
| refit.ssrhe.hessian.fit . . . . .                     | 108 |
| refit.ssrhe.hessian.l1.fit . . . . .                  | 109 |
| select.masked.graph.field . . . . .                   | 111 |
| slpl.tf.operator . . . . .                            | 112 |
| smoother.matrix . . . . .                             | 114 |
| smoother.matrix.lps . . . . .                         | 115 |
| smoother.matrix.malps . . . . .                       | 117 |
| ssrhe.hessian.operator . . . . .                      | 118 |
| ssrhe.support.grid . . . . .                          | 122 |
| summary.harmonic_smoother . . . . .                   | 123 |
| support.masked.graph.field . . . . .                  | 124 |
| synthetic.dataset.checksum . . . . .                  | 125 |
| synthetic.registry.ids . . . . .                      | 125 |
| synthetic.registry.seed . . . . .                     | 126 |
| synthetic.registry.spec . . . . .                     | 126 |
| synthetic.response.bernoulli . . . . .                | 127 |
| synthetic.response.clustered.gaussian . . . . .       | 128 |
| synthetic.response.gaussian . . . . .                 | 128 |
| synthetic.response.heteroskedastic.gaussian . . . . . | 129 |
| synthetic.response.laplace.outlier . . . . .          | 129 |

|                                              |     |
|----------------------------------------------|-----|
| synthetic.sampling.stratified . . . . .      | 130 |
| synthetic.spec . . . . .                     | 131 |
| synthetic.truth.gaussian.mixture . . . . .   | 132 |
| synthetic.truth.logit . . . . .              | 133 |
| synthetic.truth.named . . . . .              | 133 |
| synthetic.truth.occupation.mixture . . . . . | 134 |
| synthetic.truth.polynomial . . . . .         | 135 |
| transported.graph.hessian.operator . . . . . | 135 |
| validate.synthetic.dataset . . . . .         | 141 |

|              |            |
|--------------|------------|
| <b>Index</b> | <b>142</b> |
|--------------|------------|

---

|                   |                          |
|-------------------|--------------------------|
| geosmooth-package | <i>geosmooth package</i> |
|-------------------|--------------------------|

---

## Description

Geometric smoothing and conditional expectation methods for coordinate data, point clouds, and weighted graphs. The package provides local polynomial, model-averaged, trend-filtering, graph low-pass, occupation-density, and Hessian-energy methods.

## Start here

Use `browseVignettes("geosmooth")` to open the installed HTML guides, including `vignette("function-guide", package = "geosmooth")`. The [online guide](#) links to the full reference. The task links below work in installed help without an internet connection.

## Finding a function

Read `vignette("function-guide", package = "geosmooth")` for a task-based guide to all exported functions, prediction limits, and examples.

**Local regression** Start with [fit.lps](#) or [fit.malps](#).

**Lifting trend filters** Use [fit.lpl.tf](#) or [fit.slpl.tf](#) for residual penalties with optional prediction synchronization.

**Graph smoothing** Use [fit.metric.graph.lowpass](#) for spectral smoothing or [fit.graph.trend.filtering](#) for graph-difference penalties.

**Hessian penalties** Use [fit.ssrhe.hessian.regression](#) for quadratic energy or [fit.ssrhe.hessian.l1.regression](#) for an absolute-value penalty.

**Probability mass** Use [fit.density](#) for support weights, [fit.subject.od](#) for repeated visits, and [normalize.density](#) to convert an existing fitted field.

**Synthetic examples** Combine components with [synthetic.spec](#) and draw data with [materialize.synthetic](#).

**Author(s)**

**Maintainer:** Pawel Gajer <pgajer@gmail.com>

Authors:

- Pawel Gajer <pgajer@gmail.com>

Other contributors:

- Gael Guennebaud (Eigen) [contributor]
- Benoit Jacob (Eigen) [contributor]
- Authors of Eigen (Authorship and copyright in the included Eigen library) [copyright holder]
- Yixuan Qiu (Spectra) [contributor, copyright holder]
- Contributors to Spectra (Copyright in the included Spectra library) [copyright holder]
- Sunil Arya (ANN) [contributor, copyright holder]
- David M. Mount (ANN) [contributor, copyright holder]
- University of Maryland (ANN) [copyright holder]

**See Also**

```
vignette("function-guide", package = "geosmooth"), vignette("synthetic-datasets", package = "geosmooth")
```

**Examples**

```
X <- matrix(seq(0, 1, length.out = 30), ncol = 1)
y <- sin(2 * pi * X[, 1])
fit <- fit.lps(X, y, support.grid = 8L, degree.grid = 1L,
              kernel.grid = "tricube", foldid = rep(1:3, 10))
head(predict(fit))
```

---

apply.metric.graph.lowpass.path

*Apply a Metric Graph Low-Pass Filter Path*

---

**Description**

Applies every requested low-pass parameter to one response or a matrix of responses using a reusable spectral basis.

**Usage**

```

apply.metric.graph.lowpass.path(
  basis,
  y,
  eta.grid,
  filter.type = c("heat_kernel", "tikhonov", "cubic_spline", "gaussian", "exponential",
    "butterworth"),
  block.size = NULL,
  truncation.tol = 1e-04,
  unresolved.action = c("warn", "error", "allow"),
  exact.zero = TRUE
)

```

**Arguments**

|                                |                                                                                                                                             |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <code>basis</code>             | A "metric.graph.lowpass.basis" object.                                                                                                      |
| <code>y</code>                 | Numeric response vector or matrix with one row per graph vertex.                                                                            |
| <code>eta.grid</code>          | Numeric filter-parameter grid.                                                                                                              |
| <code>filter.type</code>       | Spectral low-pass filter family.                                                                                                            |
| <code>block.size</code>        | Optional number of response columns processed together.                                                                                     |
| <code>truncation.tol</code>    | Positive tolerance smaller than one used to classify truncated-basis candidates as spectrally resolved.                                     |
| <code>unresolved.action</code> | Action when a truncated-basis candidate exceeds <code>truncation.tol</code> : "warn", "error", or "allow".                                  |
| <code>exact.zero</code>        | Logical. For heat filtering, return the input exactly at $\eta = 0$ . This avoids treating a truncated spectral projection as the identity. |

**Value**

A list of class "metric.graph.lowpass.path". For one response, `fitted.values` is an  $N$  by  $J$  matrix. For multiple responses it is an  $N$  by  $J$  by  $S$  array.

**Examples**

```

adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
lengths <- list(1, c(1, 1), c(1, 1), 1)
basis <- metric.graph.lowpass.basis(adj, lengths, n.eigenpairs = 4L,
  eigen.solver = "dense")
apply.metric.graph.lowpass.path(basis, 1:4, eta.grid = c(0, 0.1, 1))

```

---

as.data.frame.synthetic\_dataset

*Summarize a synthetic dataset as one metadata row*


---

## Description

Summarize a synthetic dataset as one metadata row

## Usage

```
## S3 method for class 'synthetic_dataset'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

## Arguments

|           |                      |
|-----------|----------------------|
| x         | A synthetic_dataset. |
| row.names | Optional row names.  |
| optional  | Ignored.             |
| ...       | Ignored.             |

## Value

A one-row data frame.

## Examples

```
x <- materialize.synthetic(synthetic.registry.spec("G1"), n = 20L, seed = 1L)
as.data.frame(x)
```

---

bootstrap.malps

*Bootstrap Uncertainty For A Fixed-Selection MALPS Fit*


---

## Description

Reuses the supports, local charts, prediction supports, averaging weights, selected degree, and local-solver controls from an existing `fit.malps` object, then repeatedly calls `refit` with bootstrap-style case weights. This is a fixed-selection uncertainty diagnostic: it measures refit variability conditional on the fitted MALPS support profile and does not rerun cross-validation or rebuild supports in each replicate.

**Usage**

```
bootstrap.malps(
  object,
  B = 200L,
  weight.type = c("bayesian", "multinomial"),
  y = NULL,
  probs = NULL,
  conf.level = 0.95,
  seed = NULL,
  max.failures = max(10L, B),
  keep.weights = FALSE,
  verbose = FALSE,
  ...
)
```

**Arguments**

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>object</code>       | A "malps" object from <code>fit.malps</code> or <code>refit</code> .                       |
| <code>B</code>            | Number of successful bootstrap replicates requested.                                       |
| <code>weight.type</code>  | Bootstrap weight generator, one of "bayesian" or "multinomial".                            |
| <code>y</code>            | Optional response vector. Defaults to <code>object\$y</code> .                             |
| <code>probs</code>        | Optional nonnegative sampling probabilities for <code>weight.type = "multinomial"</code> . |
| <code>conf.level</code>   | Pointwise interval level used in the returned summary.                                     |
| <code>seed</code>         | Optional integer seed for reproducible bootstrap weights.                                  |
| <code>max.failures</code> | Maximum failed refit attempts allowed before stopping.                                     |
| <code>keep.weights</code> | Logical; store the generated bootstrap weights in the returned object.                     |
| <code>verbose</code>      | Logical; emit progress messages every ten successful replicates.                           |
| <code>...</code>          | Reserved for future extensions.                                                            |

**Details**

Two weight generators are available. "bayesian" draws positive exponential weights and rescales them to have mean one. This is the default because it preserves positive local support weights and therefore avoids many local-support degeneracies. "multinomial" draws ordinary nonparametric bootstrap counts with total count `nrow(object$X)`; this can fail when a replicate assigns zero weight to all observations in a local support, and such failures are recorded.

**Value**

A list of class "malps\_bootstrap" containing:

- `replicate.fitted.values`: an  $n \times B$  matrix of successful replicate fitted values;
- `replicate.weights`: an optional  $n \times B$  matrix of bootstrap weights when `keep.weights = TRUE`;
- `failures`: a data frame with columns `attempt`, `failure`, and `message`;
- `summary`: pointwise fitted values, bootstrap summaries, and empirical lower/upper intervals;
- bootstrap metadata such as `B.completed`, `n.failures`, `conf.level`, and `weight.type`.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.malps(X, X[, 1]^2, degree = 1L,
               support.type = "knn", support.size = 8L)
bootstrap.malps(fit, B = 5L, seed = 1L)
```

---

```
compare.masked.graph.fields
```

*Compare Masked Graph Fields with Independent Graph Workers*

---

**Description**

Runs one candidate graph per process, with dynamically scheduled jobs and no nested fold parallelism. On Unix, forked workers inherit the loaded package sources. Run from Rscript, not a GUI or multithreaded parent process.

**Usage**

```
compare.masked.graph.fields(operators, studies, workers = 1L, ...)
```

**Arguments**

|           |                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------|
| operators | Named list of metric graph low-pass operators.                                                                                       |
| studies   | Named list of argument lists containing vertex, y and group.                                                                         |
| workers   | Number of graph workers; choose using available memory and CPU. Defaults to serial execution. Windows currently requires one worker. |
| ...       | Additional arguments passed to cv.masked.graph.field.                                                                                |

**Value**

Named graph results, each containing named study results or an explicit error record. Retains per-observation predictions for common-support scoring.

---

```
compare.synthetic.dataset
```

*Compare two synthetic datasets scientifically*

---

**Description**

Compare two synthetic datasets scientifically

**Usage**

```
compare.synthetic.dataset(x, y, tolerance = c(1e-12, 1e-10))
```

**Arguments**

|           |                                              |
|-----------|----------------------------------------------|
| x, y      | Synthetic datasets.                          |
| tolerance | Absolute-plus-relative comparison tolerance. |

**Value**

A list with equality and mismatch details.

**Examples**

```
spec <- synthetic.registry.spec("G1")
x <- materialize.synthetic(spec, n = 20L, seed = 1L)
y <- materialize.synthetic(spec, n = 20L, seed = 1L)
compare.synthetic.dataset(x, y)
```

---

cv.masked.graph.field *Participant-Grouped Nested Cross-Validation for a Masked Graph Field*

---

**Description**

Select smoothing within outer training folds, predict held-out groups, and finally tune and refit on all observations. Graph construction is label-free and fixed (transductive prediction). Unsupported predictions remain NA.

**Usage**

```
cv.masked.graph.field(
  operator,
  vertex,
  y,
  group,
  lambdas = 10^seq(-6, 2, length.out = 17),
  outer.folds = 5L,
  inner.folds = 4L,
  seed = 1L,
  fold.id = NULL,
  progress = NULL
)
```

**Arguments**

|          |                                                                    |
|----------|--------------------------------------------------------------------|
| operator | An unnormalized metric.graph.lowpass.operator.                     |
| vertex   | Integer vertex index for each observed response.                   |
| y        | Finite numeric observed responses (no missing-value placeholders). |
| group    | Participant or leakage-block identifier per observation.           |

|                          |                                                                                                                          |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------|
| lambdas                  | Positive finite candidate penalties.                                                                                     |
| outer.folds, inner.folds | Numbers of grouped folds.                                                                                                |
| seed                     | Fold seed, independent of worker scheduling.                                                                             |
| fold.id                  | Optional preassigned outer fold per observation; a group must occupy exactly one fold.                                   |
| progress                 | Optional function called after each outer fold with fold and selected penalty; intended for experiment progress records. |

### Value

Outer predictions, baseline predictions, scores on supported cases, coverage, tuning records, fold assignments and final full-data field. Scores do not account for selecting a graph after comparing graphs.

---

|                  |                                    |
|------------------|------------------------------------|
| fit.chart.kernel | <i>Fit a Chart-Kernel Smoother</i> |
|------------------|------------------------------------|

---

### Description

Fits a local chart-kernel field by evaluating a Nadaraya–Watson style smoother at each evaluation point. The function is a general fitted-field model, not an occupation-density wrapper: density workflows should call [normalize.density](#) on the returned fit.

### Usage

```
fit.chart.kernel(
  X,
  y,
  X.eval = NULL,
  support.size = min(15L, nrow(X)),
  kernel = c("gaussian", "tricube", "epanechnikov", "triangular"),
  bandwidth.multiplier = 1,
  support.grid = NULL,
  kernel.grid = NULL,
  bandwidth.multiplier.grid = NULL,
  foldid = NULL,
  cv.folds = 5L,
  cv.seed = 1L,
  coordinate.method = c("coordinates", "local.pca"),
  chart.dim = NULL,
  chart.dim.grid = NULL,
  selection.strategy = c("grid", "sparse_kd", "plateau_kd"),
  chart.dim.max = NULL,
  geometry.margin = 0L,
  auto.chart.support.metric = c("coordinates", "operator", "both"),
```

```

    auto.chart.selection.metric = c("coordinates", "operator"),
    quadrature.weights = NULL,
    denominator.floor = sqrt(.Machine$double.eps),
    return.details = TRUE
)

```

## Arguments

|                                        |                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                         | Numeric matrix with one row per source/support point.                                                                                                                                                                                                                                                                                                                                                           |
| <code>y</code>                         | Numeric response or mass vector of length <code>nrow(X)</code> .                                                                                                                                                                                                                                                                                                                                                |
| <code>X.eval</code>                    | Optional numeric matrix of evaluation points. Defaults to <code>X</code> .                                                                                                                                                                                                                                                                                                                                      |
| <code>support.size</code>              | Number of source points in each local support.                                                                                                                                                                                                                                                                                                                                                                  |
| <code>kernel</code>                    | Kernel name. Supported values are "gaussian", "tricube", "epanechnikov", and "triangular".                                                                                                                                                                                                                                                                                                                      |
| <code>bandwidth.multiplier</code>      | Positive multiplier applied to the local support radius.                                                                                                                                                                                                                                                                                                                                                        |
| <code>support.grid</code>              | Optional integer candidate neighborhood sizes. If supplied, or if <code>foldid</code> is supplied, the function performs row-wise cross-validation and refits the selected candidate on all rows.                                                                                                                                                                                                               |
| <code>kernel.grid</code>               | Optional kernel candidates for cross-validation.                                                                                                                                                                                                                                                                                                                                                                |
| <code>bandwidth.multiplier.grid</code> | Optional bandwidth-multiplier candidates for cross-validation.                                                                                                                                                                                                                                                                                                                                                  |
| <code>foldid</code>                    | Optional positive integer vector assigning source rows to cross-validation folds.                                                                                                                                                                                                                                                                                                                               |
| <code>cv.folds</code>                  | Number of folds used when <code>foldid</code> is not supplied.                                                                                                                                                                                                                                                                                                                                                  |
| <code>cv.seed</code>                   | Random seed used to generate folds when <code>foldid</code> is not supplied.                                                                                                                                                                                                                                                                                                                                    |
| <code>coordinate.method</code>         | Local coordinate method. "coordinates" uses centered ambient coordinates. "local.pca" projects centered support points onto a local PCA basis.                                                                                                                                                                                                                                                                  |
| <code>chart.dim</code>                 | Local PCA dimension when <code>coordinate.method = "local.pca"</code> . If <code>NULL</code> , the dimension is <code>min(ncol(X), support.size - 1)</code> . The deployable input-only policies "auto" and "local.auto" use the same local-PCA dimension diagnostics as <a href="#">fit.lps</a> ; "auto" resolves one global chart dimension, while "local.auto" resolves one dimension per evaluation anchor. |
| <code>chart.dim.grid</code>            | Optional candidate chart dimensions for experimental row-wise cross-validation. Numeric grids can be evaluated with <code>selection.strategy = "sparse_kd"</code> .                                                                                                                                                                                                                                             |
| <code>selection.strategy</code>        | Candidate-selection strategy. "grid" evaluates the requested candidate grid. "sparse_kd" evaluates a sparse support-size by chart-dimension skeleton when <code>chart.dim.grid</code> is supplied. "plateau_kd" uses the same geometry-only support-size and chart-dimension plateau rule as <a href="#">fit.lps</a> .                                                                                          |
| <code>chart.dim.max</code>             | Optional explicit maximum chart dimension for the sparse coupled candidate family.                                                                                                                                                                                                                                                                                                                              |

`geometry.margin` Nonnegative integer subtracted from the local-PCA rank  $\text{cap}(\min(\text{ncol}(X), \text{support.size} - 1))$ . The default zero applies only the Nadaraya–Watson chart-geometry requirement; this is not a local polynomial design margin.

`auto.chart.support.metric` Support system used by `chart.dim = "auto"` or `"local.auto"`. Chart-kernel smoothers currently use coordinate supports for both coordinate and operator diagnostics.

`auto.chart.selection.metric` Which auto chart-dimension diagnostic to use when both coordinate and operator summaries are requested.

`quadrature.weights` Optional positive reference-measure weights `q_i`. Defaults to unit weights.

`denominator.floor` Positive floor used when the local denominator is numerically zero.

`return.details` Logical; if TRUE, keep per-evaluation diagnostics.

### Details

For an evaluation point  $x_u$ , the method chooses a local support  $U_u$ , builds either centered ambient coordinates or a local PCA chart, and computes

$$\hat{f}(x_u) = \frac{\sum_{i \in U_u} y_i K_h(z_{ui})}{\sum_{i \in U_u} q_i K_h(z_{ui})}.$$

Here  $z_{ui}$  is the local chart coordinate of  $x_i - x_u$ ,  $q_i$  is an optional quadrature weight, and  $K_h$  is the selected kernel.

### Value

A list with class `"chart_kernel"` containing `fitted.values`, source/evaluation supports, selected controls, and denominator diagnostics.

### Examples

```
X <- matrix(seq(0, 1, length.out = 21), ncol = 1)
y <- exp(-20 * (X[, 1] - 0.5)^2)
fit.chart.kernel(X, y, support.size = 7L, kernel = "gaussian")
```

---

fit.density

*Fit a Density Estimator*


---

### Description

Dispatches to a dedicated density estimator. Subject-occupation density estimation is one application: construct a sparse mass vector over a fixed support set and call this generic density layer.

**Usage**

```
fit.density(
  X,
  weights = NULL,
  method = c("empirical", "graph_random_walk"),
  graph = NULL,
  graph.control = list(),
  density.control = list(),
  return.details = TRUE,
  ...
)
```

**Arguments**

|                              |                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>               | Numeric matrix with one row per support point.                                                                                                                                                                                                                                                                                                           |
| <code>weights</code>         | Optional nonnegative mass/count vector of length <code>nrow(X)</code> . Required by count-based density methods.                                                                                                                                                                                                                                         |
| <code>method</code>          | Density method identifier.                                                                                                                                                                                                                                                                                                                               |
| <code>graph</code>           | Optional precomputed graph object.                                                                                                                                                                                                                                                                                                                       |
| <code>graph.control</code>   | Named list for random walks: <code>walk.steps</code> (or <code>walk.step</code> ), <code>affinity.method</code> , <code>affinity.scale</code> , <code>affinity.epsilon</code> , and logical <code>normalize</code> . Underscore aliases are accepted, but conflicting aliases and unknown names are rejected. Must be empty for the empirical estimator. |
| <code>density.control</code> | List controlling clipping, normalization, and accounting checks. Recognized entries are <code>mass.tol</code> , <code>neg.tol</code> , logical <code>clip.negative</code> and <code>renormalize</code> , and diagnostic controls <code>smoothness.adj.list</code> and logical <code>smoothness.auto.1d</code> . Unknown or duplicated names are errors.  |
| <code>return.details</code>  | Logical; if <code>TRUE</code> , keep diagnostic details in the result.                                                                                                                                                                                                                                                                                   |
| <code>...</code>             | Reserved for future methods; unused arguments are errors.                                                                                                                                                                                                                                                                                                |

**Value**

A list of class "density\_fit" with fields `method.id`, `status`, `rho`, `empirical.rho`, `fitted.raw`, `theta`, `accounting`, `smoothness`, `timing`, `diagnostics`, and `warnings`.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 6), ncol = 1)
fit.density(X, weights = c(1, 0, 2, 0, 0, 1), method = "empirical")
```

---

fit.graph.trend.filtering

*Fit Graph Trend Filtering*


---

## Description

Fits graph trend filtering on a supplied undirected graph. Supported orders are 0L, 1L, and 2L:

$$\hat{\beta}_\lambda = \arg \min_{\beta \in \mathbb{R}^n} \left\{ \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\Delta_w^{(k+1)} \beta\|_1 \right\}.$$

## Usage

```
fit.graph.trend.filtering(
  adj.list,
  weight.list = NULL,
  y,
  order = 0L,
  lambda.grid = NULL,
  lambda.selection = c("cv", "fixed"),
  weight.rule = c("conductance", "sqrt.conductance", "unit"),
  operator.family = c("graph.laplacian.recursive", "path.divided.difference"),
  path.family = c("branch.continuation", "all.simple"),
  path.weighting = c("unit", "anchor.mean"),
  n.lambda = 40L,
  nfolds = 5L,
  foldid = NULL,
  maxsteps = 2000L,
  minlam = 0,
  approx = FALSE,
  rtol = 1e-07,
  btol = 1e-07,
  eps = 1e-04,
  verbose = FALSE
)
```

## Arguments

|             |                                                                                                                                                                                                                                                                                                                           |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| adj.list    | List of integer neighbor vectors using 1-based vertex indices. The graph must be undirected: every edge must appear in both endpoint adjacency lists.                                                                                                                                                                     |
| weight.list | Optional list of positive edge weights parallel to adj.list. For operator.family = "graph.laplacian.recursive", weights are graph trend-filtering weights or conductances. For operator.family = "path.divided.difference", weights are metric edge lengths used to define path coordinates. If NULL, all values are one. |
| y           | Numeric response vector of length length(adj.list).                                                                                                                                                                                                                                                                       |

|                  |                                                                                                                                                                                                                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| order            | Integer trend-filtering order. Supported values are 0L, 1L, and 2L.                                                                                                                                                                                                                                             |
| lambda.grid      | Optional non-negative lambda grid. For lambda.selection = "fixed", this must contain exactly one value. For lambda.selection = "cv", NULL builds a default grid from the fitted full-data solution path.                                                                                                        |
| lambda.selection | "cv" for vertex K-fold cross-validation or "fixed" for a supplied fixed lambda.                                                                                                                                                                                                                                 |
| weight.rule      | Character scalar. "conductance" uses weight.list directly as $\omega_e$ ; "sqrt.conductance" uses $\sqrt{w_e}$ ; "unit" ignores weight.list. This applies to operator.family = "graph.laplacian.recursive".                                                                                                     |
| operator.family  | Character scalar selecting the penalty operator family. The default "graph.laplacian.recursive" preserves the existing graph trend-filtering semantics. "path.divided.difference" uses local pathwise finite differences and is designed to reproduce classical one-dimensional trend filtering on path graphs. |
| path.family      | Character scalar used for operator.family = "path.divided.difference". "branch.continuation" keeps paths whose internal vertices do not pass through branch points. "all.simple" keeps all simple paths of the required length.                                                                                 |
| path.weighting   | Character scalar used for operator.family = "path.divided.difference". "unit" assigns every canonical path row weight one. "anchor.mean" divides rows by the number of retained canonical paths sharing the same canonical start vertex.                                                                        |
| n.lambda         | Number of default lambda candidates when lambda.grid = NULL and lambda.selection = "cv".                                                                                                                                                                                                                        |
| nfolds           | Number of CV folds.                                                                                                                                                                                                                                                                                             |
| foldid           | Optional integer fold assignments of length length(y).                                                                                                                                                                                                                                                          |
| maxsteps         | Maximum number of path steps passed to <b>genlasso</b> .                                                                                                                                                                                                                                                        |
| minlam           | Minimum lambda passed to <b>genlasso</b> .                                                                                                                                                                                                                                                                      |
| approx           | Logical; passed to <b>genlasso</b> .                                                                                                                                                                                                                                                                            |
| rtol, btol, eps  | Numerical controls passed to <b>genlasso</b> .                                                                                                                                                                                                                                                                  |
| verbose          | Logical. If TRUE, pass verbose output to <b>genlasso</b> .                                                                                                                                                                                                                                                      |

## Details

Graph trend filtering is intended as an  $\ell_1$ -adaptive comparator to the  $\ell_2$  graph smoothers now housed in **geosmooth** and the legacy rdgraph comparator. The low-pass smoother penalizes broad quadratic graph roughness through a term like

$$\eta f^\top L f,$$

which shrinks high-frequency variation everywhere. Graph trend filtering instead penalizes absolute graph differences. For order = 0L, this is

$$\lambda \sum_e \omega_e |\beta_j - \beta_i|,$$

so many edge differences can become exactly zero while selected edges carry jumps. In this sense order = 0L is locally adaptive: fitted values can be piecewise constant over graph regions. Higher orders use

$$\Delta_w^{(1)} = D_w, \quad \Delta_w^{(2)} = L_w, \quad \Delta_w^{(3)} = D_w L_w,$$

with  $L_w = D_w^\top D_w$ .

The supplied `weight.list`, when used, is interpreted as a graph trend-filtering edge weight or conductance for `operator.family = "graph.laplacian.recursive"`. For `operator.family = "path.divided.difference"`, `weight.list` is interpreted as metric edge length and is used to form local path coordinates. Use `weight.rule = "sqrt.conductance"` when the recursive graph square-root convention is desired, and `weight.rule = "unit"` for an unweighted recursive graph fused-lasso penalty.

The solver backend is currently **genlasso**. Cross-validation refits the generalized-lasso problem on vertex-held-out training sets using a selection matrix  $X$ ; this is useful for small and moderate diagnostic problems but is not yet a scalable production graph-trend-filtering solver.

### Value

A list of class `"graph.trend.filtering.fit"` containing fitted values, residuals, selected lambda, the operator, path metadata, and CV diagnostics when requested.

### References

Wang, Y.-X., Sharpnack, J., Smola, A., and Tibshirani, R. J. (2016). Trend filtering on graphs. *Journal of Machine Learning Research*, 17(105), 1–41. <https://www.jmlr.org/papers/v17/15-147.html>

### Examples

```
if (requireNamespace("genlasso", quietly = TRUE)) {
  adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
  fit.graph.trend.filtering(
    adj, y = c(0, 0.2, 1.8, 2), lambda.grid = 0.3,
    lambda.selection = "fixed", weight.rule = "unit"
  )
}
```

---

`fit.local.likelihood`    *Fit a Local-Likelihood Density or Bernoulli Smoother*

---

### Description

Fits a local likelihood model at each evaluation point and reads off one raw fitted value at that evaluation anchor. Density and Bernoulli workflows should convert the returned fitted field with `normalize.density` when the target is a subject-occupation density.

### Usage

```
fit.local.likelihood(
  X,
  y,
  X.eval = NULL,
```

```

likelihood.family = c("density", "bernoulli"),
support.size = min(15L, nrow(X)),
degree = 1L,
kernel = c("gaussian", "tricube", "epanechnikov", "triangular"),
bandwidth.multiplier = 1,
support.grid = NULL,
degree.grid = NULL,
kernel.grid = NULL,
bandwidth.multiplier.grid = NULL,
lambda.ridge.grid = NULL,
foldid = NULL,
cv.folds = 5L,
cv.seed = 1L,
coordinate.method = c("coordinates", "local.pca"),
chart.dim = NULL,
chart.dim.grid = NULL,
selection.strategy = c("grid", "sparse_kd", "plateau_kd"),
chart.dim.max = NULL,
design.margin = 2L,
auto.chart.support.metric = c("coordinates", "operator", "both"),
auto.chart.selection.metric = c("coordinates", "operator"),
quadrature.weights = NULL,
lambda.ridge = 1e-08,
min.local.mass = sqrt(.Machine$double.eps),
min.nonzero.mass = 1L,
fallback = c("degree0", "zero", "chart_kernel", "na"),
optimizer = c("newton", "optim"),
max.iter = 50L,
tol = 1e-08,
return.details = TRUE
)

```

## Arguments

|                                |                                                                                                                                                                                                                                                                            |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                 | Numeric matrix with one row per source/support point.                                                                                                                                                                                                                      |
| <code>y</code>                 | Numeric response vector. For <code>likelihood.family = "density"</code> , this must be a nonnegative mass/intensity vector with positive total mass. For <code>"bernoulli"</code> , values must lie in $[0, 1]$ .                                                          |
| <code>X.eval</code>            | Optional numeric matrix of evaluation points. Defaults to <code>X</code> .                                                                                                                                                                                                 |
| <code>likelihood.family</code> | Local likelihood family.                                                                                                                                                                                                                                                   |
| <code>support.size</code>      | Number of source points in each local support.                                                                                                                                                                                                                             |
| <code>degree</code>            | Local chart feature degree. Supported values are 0, 1, and 2. The density branch omits an intercept because the intercept is not identifiable in the normalized local density likelihood. The Bernoulli branch includes an intercept in its internal logistic feature map. |
| <code>kernel</code>            | Kernel name. Supported values are <code>"gaussian"</code> , <code>"tricube"</code> , <code>"epanechnikov"</code> , and <code>"triangular"</code> .                                                                                                                         |

|                             |                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bandwidth.multiplier        | Positive multiplier applied to the local support radius.                                                                                                                                                                                                                                                                                                              |
| support.grid                | Optional integer candidate neighborhood sizes for cross-validation. CV selection is currently implemented for likelihood.family = "bernoulli".                                                                                                                                                                                                                        |
| degree.grid                 | Optional local polynomial degree candidates for Bernoulli cross-validation.                                                                                                                                                                                                                                                                                           |
| kernel.grid                 | Optional kernel candidates for Bernoulli cross-validation.                                                                                                                                                                                                                                                                                                            |
| bandwidth.multiplier.grid   | Optional bandwidth-multiplier candidates for Bernoulli cross-validation.                                                                                                                                                                                                                                                                                              |
| lambda.ridge.grid           | Optional ridge-penalty candidates for Bernoulli cross-validation.                                                                                                                                                                                                                                                                                                     |
| foldid                      | Optional positive integer vector assigning source rows to cross-validation folds.                                                                                                                                                                                                                                                                                     |
| cv.folds                    | Number of folds used when foldid is not supplied.                                                                                                                                                                                                                                                                                                                     |
| cv.seed                     | Random seed used to generate folds when foldid is not supplied.                                                                                                                                                                                                                                                                                                       |
| coordinate.method           | Local coordinate method. "coordinates" uses centered ambient coordinates. "local.pca" projects centered support points onto a local PCA basis.                                                                                                                                                                                                                        |
| chart.dim                   | Local PCA dimension when coordinate.method = "local.pca". If NULL, the dimension is min(ncol(X), support.size - 1). The deployable input-only policies "auto" and "local.auto" use the same local-PCA dimension diagnostics as <a href="#">fit.lps</a> ; "auto" resolves one global chart dimension, while "local.auto" resolves one dimension per evaluation anchor. |
| chart.dim.grid              | Optional candidate chart dimensions for experimental cross-validation. Numeric grids can be evaluated with selection.strategy = "sparse_kd".                                                                                                                                                                                                                          |
| selection.strategy          | Candidate-selection strategy. "grid" evaluates the requested candidate grid. "sparse_kd" evaluates a sparse support-size by chart-dimension skeleton when chart.dim.grid is supplied. "plateau_kd" uses the same geometry-only support-size and chart-dimension plateau rule as <a href="#">fit.lps</a> .                                                             |
| chart.dim.max               | Optional explicit maximum chart dimension for the sparse coupled candidate family.                                                                                                                                                                                                                                                                                    |
| design.margin               | Nonnegative integer feasibility margin used to screen local polynomial design size before candidate evaluation.                                                                                                                                                                                                                                                       |
| auto.chart.support.metric   | Support system used by chart.dim = "auto" or "local.auto". Local-likelihood smoothers currently use coordinate supports for both coordinate and operator diagnostics.                                                                                                                                                                                                 |
| auto.chart.selection.metric | Which auto chart-dimension diagnostic to use when both coordinate and operator summaries are requested.                                                                                                                                                                                                                                                               |
| quadrature.weights          | Optional positive reference-measure weights. Defaults to unit weights.                                                                                                                                                                                                                                                                                                |
| lambda.ridge                | Nonnegative ridge penalty on identifiable coefficients.                                                                                                                                                                                                                                                                                                               |
| min.local.mass              | Minimum local kernel-weighted mass needed before attempting a higher-degree density fit. For Bernoulli fits this is used only as diagnostic telemetry.                                                                                                                                                                                                                |

|                               |                                                                                                                    |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>min.nonzero.mass</code> | Minimum number of locally weighted positive-mass source points needed before attempting a higher-degree local fit. |
| <code>fallback</code>         | Fallback policy for underidentified or failed higher-degree local fits. Zero local mass always uses "zero".        |
| <code>optimizer</code>        | Optimizer for nonzero-degree density fits. "newton" is implemented; "optim" is accepted and delegates to BFGS.     |
| <code>max.iter</code>         | Maximum optimizer iterations.                                                                                      |
| <code>tol</code>              | Convergence tolerance for gradient norm and step norm.                                                             |
| <code>return.details</code>   | Logical; if TRUE, keep per-evaluation diagnostics.                                                                 |

### Details

The `likelihood.family = "density"` branch uses a local exponential tilt of the chart reference measure. The `"bernoulli"` branch uses a weighted local logistic likelihood and returns fitted probabilities at the evaluation anchors.

### Value

A list with class `"local_likelihoood"` containing `fitted.values`, selected controls, and local solver diagnostics.

### Examples

```
X <- matrix(seq(0, 1, length.out = 21), ncol = 1)
y <- exp(-20 * (X[, 1] - 0.5)^2)
y <- y / sum(y)
fit.local.likelihood(X, y, likelihood.family = "density",
  support.size = 7L, degree = 0L)
```

---

fit.lpl.tf

*Fit A Local Polynomial Lifting Trend Filter*


---

### Description

Fits an LPL-TF estimator with a fixed operator:

$$\hat{f} = \arg \min_f \frac{1}{2} \|y - f\|_2^2 + \lambda \|A_{\text{LPL}} f\|_1.$$

### Usage

```
fit.lpl.tf(
  X = NULL,
  y,
  operator = NULL,
  adj.list = NULL,
```

```

weight.list = NULL,
graph = NULL,
degree = 2L,
lambda.grid = NULL,
lambda = NULL,
lambda.selection = c("cv", "fixed"),
operator.grid = NULL,
foldid = NULL,
cv.folds = 5L,
cv.loss = c("mse", "rmse", "mae"),
cv.seed = NULL,
cv.repeats = 1L,
solver = c("genlasso"),
selection = c("min", "one.se"),
n.lambda = 80L,
maxsteps = 2000L,
minlam = 0,
approx = FALSE,
rtol = 1e-07,
btol = 1e-07,
eps = 1e-04,
verbose = FALSE,
...
)

```

## Arguments

|                                           |                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                            | Optional coordinate matrix used to build <code>lpl.tf.operator</code> when operator is NULL.                                                                                                                                                                                                                                                                                                              |
| <code>y</code>                            | Numeric response vector.                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>operator</code>                     | Optional "lpl_tf_operator" object.                                                                                                                                                                                                                                                                                                                                                                        |
| <code>adj.list, weight.list, graph</code> | Optional graph inputs passed to <code>lpl.tf.operator</code> when operator is NULL.                                                                                                                                                                                                                                                                                                                       |
| <code>degree</code>                       | Polynomial degree passed to <code>lpl.tf.operator</code> when operator is NULL.                                                                                                                                                                                                                                                                                                                           |
| <code>lambda.grid</code>                  | Optional nonnegative lambda grid.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>lambda</code>                       | Optional fixed lambda shortcut. If supplied, it is used as the single fixed lambda and <code>lambda.selection</code> must be "fixed".                                                                                                                                                                                                                                                                     |
| <code>lambda.selection</code>             | "cv" or "fixed". CV requires an explicit <code>lambda.grid</code> so the candidate grid is not generated from the full response vector before fold scoring.                                                                                                                                                                                                                                               |
| <code>operator.grid</code>                | Optional operator candidate grid. Supply a data frame with one row per candidate or a list of named lists. Candidate fields may include operator-construction arguments such as <code>degree</code> , <code>support.type</code> , <code>support.size</code> , <code>min.support</code> , <code>support.buffer</code> , <code>kernel</code> , <code>row.normalize</code> , and <code>local.solver</code> . |
| <code>foldid</code>                       | Optional deterministic fold assignments for CV.                                                                                                                                                                                                                                                                                                                                                           |
| <code>cv.folds</code>                     | Number of generated folds when <code>foldid = NULL</code> .                                                                                                                                                                                                                                                                                                                                               |

|                                           |                                                                                                                                    |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| cv.loss                                   | Cross-validation loss, "mse", "rmse", or "mae". "rmse" uses the same lambda ordering as MSE and reports square-rooted fold errors. |
| cv.seed                                   | Reserved for reproducible generated folds. Generated folds are deterministic even when this is NULL.                               |
| cv.repeats                                | Number of CV repeats. The current implementation supports only one repeat.                                                         |
| solver                                    | Solver backend. The current implementation supports only "genlasso".                                                               |
| selection                                 | Lambda selection rule, "min" or "one.se".                                                                                          |
| n.lambda                                  | Number of generated lambdas when a genlasso path is used and lambda.grid = NULL.                                                   |
| maxsteps, minlam, approx, rtol, btol, eps | Genlasso controls.                                                                                                                 |
| verbose                                   | Logical.                                                                                                                           |
| ...                                       | Additional arguments passed to <a href="#">lpl.tf.operator</a> when the operator is built internally.                              |

**Value**

A list of class "lpl\_tf".

**Examples**

```
if (requireNamespace("genlasso", quietly = TRUE)) {
  X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
  fit.lpl.tf(X, sin(2 * pi * X[, 1]), degree = 1L,
    support.type = "knn", support.size = 7L,
    lambda = 0.1, lambda.selection = "fixed")
}
```

---

fit.lps

---

*Fit a Local Polynomial Smoother*


---

**Description**

Fits a local polynomial smoother (LPS) and selects its support size, polynomial degree, and kernel by cross-validation. By default, the smoother works in the observed ambient coordinates: each prediction point uses its nearest training points in Euclidean distance, centers the support at the prediction point, fits a weighted local polynomial, and uses the fitted intercept as the prediction.

**Usage**

```
fit.lps(
  X,
  y,
  foldid = NULL,
  support.grid = c(10L, 15L, 20L),
  degree.grid = 0:2,
```

```

kernel.grid = c("gaussian", "tricube"),
cv.folds = 5L,
cv.seed = 1L,
X.eval = NULL,
coordinate.method = c("coordinates", "local.pca"),
chart.dim = NULL,
chart.dim.grid = NULL,
local.chart.method = c("pca", "second.order.svd"),
auto.chart.support.metric = c("coordinates", "operator", "both"),
auto.chart.selection.metric = c("coordinates", "operator"),
backend = c("auto", "R", "cpp", "cpp.local.pca"),
design.basis = c("orthogonal.polynomial.drop", "monomial", "weighted.qr",
  "weighted.qr.drop"),
design.drop.tol = 1e-08,
ridge.multiplier.grid = c(0, 1e-10, 1e-08),
ridge.condition.max = 1e+12,
unstable.action = c("na", "mean"),
outcome.family = c("gaussian", "bernoulli", "binomial"),
bandwidth.multiplier.grid = 1,
keep.cv.predictions = FALSE,
ridge.shrinkage.target = c("zero", "local.mean"),
selection.strategy = c("grid", "sparse_kd", "plateau_kd"),
chart.activation = c("none", "subject.od"),
chart.activation.response = NULL,
chart.activation.control = list(),
chart.dim.max = NULL,
design.margin = 2L
)

```

## Arguments

|                                |                                                                                                                                                                                                 |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                 | Numeric design/coordinate matrix with one observation per row.                                                                                                                                  |
| <code>y</code>                 | Numeric response vector with length <code>nrow(X)</code> .                                                                                                                                      |
| <code>foldid</code>            | Optional positive integer vector assigning rows to CV folds.                                                                                                                                    |
| <code>support.grid</code>      | Integer candidate neighborhood sizes.                                                                                                                                                           |
| <code>degree.grid</code>       | Integer polynomial degrees. Currently degrees 0, 1, and 2 are supported.                                                                                                                        |
| <code>kernel.grid</code>       | Candidate kernels. Supported kernels are "gaussian", "tricube", "epanechnikov", and "triangular".                                                                                               |
| <code>cv.folds</code>          | Number of folds used when <code>foldid</code> is not supplied.                                                                                                                                  |
| <code>cv.seed</code>           | Random seed used to generate folds when <code>foldid</code> is not supplied.                                                                                                                    |
| <code>X.eval</code>            | Optional matrix of prediction locations. Defaults to <code>X</code> .                                                                                                                           |
| <code>coordinate.method</code> | Local coordinate system. "coordinates" uses centered ambient coordinates. "local.pca" uses a local PCA chart.                                                                                   |
| <code>chart.dim</code>         | Chart dimension for <code>coordinate.method = "local.pca"</code> . If <code>NULL</code> , defaults to <code>ncol(X)</code> . The special value "auto" estimates one global chart dimension from |

|                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | observed $X$ only. The experimental special value "local.auto" estimates a local chart dimension separately for each prediction anchor in the ordinary local-PCA R backend.                                                                                                                                                                                                                                                                                                       |
| chart.dim.grid              | Optional candidate chart dimensions for experimental coupled support-size by chart-dimension selection. When supplied, coordinate.method must be "local.pca" and the evaluated candidates use scalar numeric chart dimensions after feasibility filtering. The default NULL preserves the historical single-chart.dim behavior.                                                                                                                                                   |
| local.chart.method          | Local chart constructor used when coordinate.method = "local.pca". "pca" preserves the ordinary local-PCA chart path. "second.order.svd" uses an experimental curvature-corrected second-order local SVD chart and records compact chart fallback diagnostics. This option is opt-in and does not affect ambient coordinate fits.                                                                                                                                                 |
| auto.chart.support.metric   | Support system used when chart.dim = "auto" or "local.auto". Included for consistency with LPL-TF and S-LPL-TF; because this smoother uses coordinate supports, "operator" is equivalent to "coordinates".                                                                                                                                                                                                                                                                        |
| auto.chart.selection.metric | Which auto chart-dimension diagnostic to select when both diagnostics are requested.                                                                                                                                                                                                                                                                                                                                                                                              |
| backend                     | Computation backend. "auto" uses the C++ backend for coordinate.method = "coordinates" and the R reference backend for coordinate.method = "local.pca". "R" always uses the reference implementation. "cpp" requires ambient coordinates. "cpp.local.pca" is an opt-in prototype backend for coordinate.method = "local.pca" with local.chart.method = "pca".                                                                                                                     |
| design.basis                | Local polynomial design backend. "monomial" uses the ordinary raw monomial design. "weighted.qr" solves the same design using an explicit weighted-QR numerical path. "weighted.qr.drop" first drops numerically dependent local design columns by weighted QR before solving. "orthogonal.polynomial.drop" replaces the local monomial design by a weighted-orthogonal basis spanning its estimable polynomial directions, dropping numerically rank-deficient directions first. |
| design.drop.tol             | Relative QR tolerance used by design.basis = "weighted.qr.drop" or design.basis = "orthogonal.polynomial.drop".                                                                                                                                                                                                                                                                                                                                                                   |
| ridge.multiplier.grid       | Nonnegative scale-relative ridge multipliers tried by the R local-solve backend. The smallest multiplier whose penalized local normal equations pass ridge.condition.max is used.                                                                                                                                                                                                                                                                                                 |
| ridge.condition.max         | Maximum allowed condition number for the penalized local normal equations. Use Inf to disable this guard.                                                                                                                                                                                                                                                                                                                                                                         |
| unstable.action             | Action when no local solve passes the rank and condition guards. "mean" preserves the historical weighted-mean fallback. "na" returns NA, causing CV candidates with unstable predictions to be avoided.                                                                                                                                                                                                                                                                          |

- `outcome.family` Response family. "gaussian" preserves the ordinary numeric-response LPS behavior. "bernoulli" treats 0/1 responses as numeric conditional-expectation targets for  $\Pr(Y = 1 \mid X)$ , keeps the same local least-squares fitting core, clips reported response-scale probabilities to  $[0, 1]$ , selects candidates by the observed CV Brier score of the clipped predictions (E2.12: the selection score is the deployed metric, which requires per-point CV predictions, so "bernoulli" – like "binomial" – always uses the R backend: `backend = "auto"` resolves to "R" and an explicit C++ backend is an error), and records Brier/log-loss probability diagnostics. "binomial" uses local weighted logistic polynomial fits, selects candidates by observed CV log loss with the log-loss probability truncation pinned at  $1e-6$  (E2.12) and with any candidate having a non-finite CV prediction scored `Inf` – unselectable, the same rule as the gaussian/bernoulli selection scores (E2.15) – and reports probability diagnostics on the fitted probabilities. The local logistic IRLS uses deviance step-halving (E2.14): a Newton update is accepted only if the weighted binomial deviance does not increase by more than  $1e-8$ ; otherwise the step is halved toward the current iterate (at most 30 times, after which the solve is declared non-convergent). The deviance is evaluated on the same  $[-35, 35]$  clamped linear predictor the IRLS update uses, so it is finite for every finite iterate. A solve that does not converge within the iteration cap (including under exact separation, where the unpenalized logistic MLE does not exist) falls back deterministically to the local weighted event rate under `unstable.action = "mean"` or to NA under `unstable.action = "na"`; every fallback is counted in `logistic.diagnostics(fallback.path, event.rate.fallback, na.failure)`.
- `bandwidth.multiplier.grid` Nonnegative bandwidth multipliers added to the CV candidate grid. For a candidate with multiplier  $b$ , the local kernel bandwidth becomes  $h = b \cdot d_{(K)}$  where  $d_{(K)}$  is the distance to the  $K$ -th nearest support neighbor, decoupling the kernel scale from the support size. The default 1 reproduces the historical behavior exactly (the bandwidth equals the support radius). Any grid other than exactly 1 requires the R reference backend: `backend = "auto"` then resolves to "R", and explicit `backend = "cpp"` or `cpp.local.pca` is an error. The selected multiplier is returned in `$selected$bandwidth.multiplier` and as a `bandwidth.multiplier` column of `cv.table`.
- `keep.cv.predictions` Logical; if TRUE, the returned object additionally carries `cv.predictions`, the matrix of out-of-fold CV predictions with one column per `cv.table` row, so selection scores can be recomputed from the actual CV predictions (E2.12). NULL on the C++ CV paths (reachable only for `outcome.family = "gaussian"`), which do not materialize per-point predictions. Default FALSE leaves the returned object exactly as before.
- `ridge.shrinkage.target` Shrinkage target of the local ridge penalty in the least-squares solve (E2.13). The default "zero" preserves the historical behavior bit-for-bit: in the orthogonal design bases the penalty acts on every transformed direction, including the constant, so a large ridge multiplier shrinks the local prediction toward 0. "local.mean" – the statistically recommended setting – leaves the constant direction unpenalized via a weighted-centering reparametrization, so a large

ridge shrinks the prediction toward the local weighted mean instead, and  $\rho = 0$  remains the unpenalized weighted least-squares solve. The two settings coincide for the non-orthogonal design bases ("monomial", "weighted.qr", "weighted.qr.drop"), whose constant column is already unpenalized, and at  $\rho = 0$ . The setting applies to the least-squares solve (outcome.family "gaussian" / "bernoulli"); it has *no effect* on the "binomial" local logistic solver, whose ridge keeps the historical structure (a warning is issued if combined).

#### selection.strategy

Candidate-selection strategy. "grid" preserves the full Cartesian candidate grid. The experimental "sparse\_kd" strategy evaluates a sparse coupled support-size by chart-dimension skeleton when chart.dim.grid is supplied. "plateau\_kd" is a geometry-only selector: from observed  $X$  only, it estimates the local PCA variance dimension over the supplied support grid, finds the initial support-size plateau where that dimension is stable from the smallest support size, aggregates plateau endpoints across representative anchors, and evaluates the resulting single (support.size, chart.dim) candidate.

#### chart.activation

Optional sparse-response chart activation rule. "none" preserves the ordinary LPS behavior. "subject.od" is intended for subject-occupation density workflows: a local chart whose support contains too little subject occupation mass, too few positive subject-visited points, or only fringe occupation receives fitted value zero without constructing the local polynomial fit.

#### chart.activation.response

Optional nonnegative response used only by chart.activation = "subject.od" to decide whether a chart is active. When omitted,  $y$  is used.

#### chart.activation.control

List controlling sparse chart activation. Supported fields are mass.min, n.positive.min, positive.tol, core.weight.rule, core.weight.quantile, and core.weight.min. The OD default is two positive support points and a chart-specific 0.25 weight quantile.

#### chart.dim.max

Optional explicit maximum chart dimension for the experimental coupled selector.

#### design.margin

Nonnegative prefit margin used to mark coupled (support.size, chart.dim) candidates infeasible when the full local polynomial design would be underdetermined.

## Details

The optional coordinate.method = "local.pca" mode keeps the same support and kernel weighting rule, but builds the local polynomial in a local PCA chart centered at each prediction point. With chart.dim = "auto", the chart dimension is estimated as one global scalar from observed  $X$  only, using the same shared local-PCA dimension helper used by LPL-TF and S-LPL-TF. The experimental chart.dim = "local.auto" mode estimates an input-only local chart dimension separately for each prediction anchor.

## Value

A list of class "lps" with response-scale `fitted.values`, unmodified local least-squares `fitted.values.raw`, selected parameters, a candidate CV table, the requested `local.chart.method`, and the effective chart method used for reporting. In `outcome.family = "bernoulli"` or `"binomial"` mode, `fitted.values` are response-scale probabilities in  $[0, 1]$ , `fitted.values.raw` are the un-clipped conditional-expectation estimates for `"bernoulli"` and the fitted probabilities for `"binomial"`, `cv.table$cv.brier.observed` is the observed CV Brier score of the response-scale (clipped) predictions with `Inf` for candidates having any non-finite prediction, and `probability.diagnostics` records raw/clipped probability ranges, out-of-range fractions, and Brier/log-loss diagnostics. The Brier and log-loss diagnostics are defined only when the fitted predictions have the same length as the training response, which is the default `X.eval = X.path`. In `"binomial"` mode, `logistic.diagnostics` records local logistic solve attempts, convergence statuses, fallback-path counts, event-rate fallback counts, and NA failure counts separately for CV and final fitting.

## Choosing controls

**First fit:** supply `X`, `y`, `foldid`, and small `support.grid`, `degree.grid`, and `kernel.grid` candidates. See the installed function guide.

**Selection:** `cv.folds`, `cv.seed`, and `selection.strategy` control the search. Candidate combinations and folds multiply the fitting work; use `lps.nested.cv()` for an outer assessment of the tuning procedure.

**Geometry:** `coordinate.method`, `chart.dim`, and local chart controls describe neighborhoods and coordinate systems, not a different response model.

**Numerics:** `backend`, `design.basis`, rank thresholds and ridge controls govern stable local solves. Inspect `lps.backend.diagnostics()` before changing them. A numerical ridge is distinct from neighborhood selection.

**Diagnostics:** `selected`, `cv.table`, and `diagnostics` retain selection and local-fit information. A dense `smoother.matrix` needs one entry per evaluation/training pair; it is restricted to supported configurations.

## References

Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and Its Applications*. Chapman and Hall/CRC. ISBN 9780412983214.

## Examples

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
y <- sin(2 * pi * X[, 1])
fit <- fit.lps(
  X, y, support.grid = 8L, degree.grid = 1L,
  kernel.grid = "tricube", cv.folds = 2L, backend = "R"
)
head(fit$fitted.values)
```

fit.malps

*Fit Model-Averaged Local Polynomial Smoothing***Description**

Fits the current model-averaged local polynomial smoother (MALPS) on a supplied coordinate matrix. MALPS fits local polynomial regressions around observed anchor points and averages all positive-weight local predictions at each training point. The current implementation supports supplied coordinates, observed or supplied coordinate anchors, fixed support parameters, deterministic cross-validation and exact dense GCV over support parameters, local PCA chart coordinates, ordinary and weighted new-response refits, coordinate-space new-point prediction, linear-smoother diagnostics, and fixed-selection bootstrap uncertainty summaries. Graph-geodesic supports are available for training and refit; graph-geodesic new-point prediction is deferred.

**Usage**

```
fit.malps(
  X,
  y,
  graph = NULL,
  adj.list = NULL,
  weight.list = NULL,
  graph.stage = "final",
  anchor.index = NULL,
  anchor.coordinates = NULL,
  degree = 2L,
  degree.grid = NULL,
  support.type = c("adaptive.radius", "knn", "fixed.radius"),
  support.size = NULL,
  support.grid = NULL,
  radius = NULL,
  radius.grid = NULL,
  min.support = NULL,
  min.support.grid = NULL,
  support.buffer = 3L,
  kernel = c("epanechnikov", "triangular", "gaussian", "tricube"),
  kernel.grid = NULL,
  model.weight.rule = c("none", "condition", "support", "boundary", "quality"),
  duplicate.action = c("keep", "error"),
  coordinate.method = c("coordinates", "local.pca"),
  chart.dim = NULL,
  support.metric = c("auto", "coordinates", "graph.geodesic"),
  auto.chart.support.metric = c("coordinates", "operator", "both"),
  auto.chart.selection.metric = c("coordinates", "operator"),
  support.selection = c("fixed", "cv", "gcv"),
  foldid = NULL,
  cv.folds = 5L,
```

```

cv.loss = c("rmse", "mae", "mse"),
cv.repeats = 1L,
cv.seed = NULL,
cv.one.se = FALSE,
gcv.exact.max.n = 1000L,
local.solver = c("auto", "normal.equations", "qr", "svd"),
normal.equations.max.condition = 1e+08,
robust.iterations = 0L,
robust.tuning.constant = 6,
verbose = FALSE,
...
)

```

### Arguments

|                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                  | Numeric coordinate matrix with one row per observation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>y</code>                  | Numeric response vector with length <code>nrow(X)</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>graph</code>              | Optional supported <b>dgraphs</b> graph object, for example from <code>dgraphs::create.rknn.graph()</code> . When <code>support.metric = "graph.geodesic"</code> or <code>"auto"</code> with a graph supplied, weighted shortest-path distances from this graph are used for support construction.                                                                                                                                                                                                                                            |
| <code>adj.list</code>           | Optional 1-based undirected adjacency list. Must be supplied together with <code>weight.list</code> when using a supplied graph payload.                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>weight.list</code>        | Optional positive edge-length list parallel to <code>adj.list</code> . Edge weights are interpreted as graph lengths.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>graph.stage</code>        | Graph lifecycle stage used when graph is a <b>dgraphs</b> graph object. Supported stages include <code>"final"</code> , <code>"raw"</code> , and <code>"pruned"</code> when those payloads are available on the supplied graph.                                                                                                                                                                                                                                                                                                               |
| <code>anchor.index</code>       | Optional integer vector of observed rows used as local model anchors. Defaults to all rows when <code>anchor.coordinates</code> is <code>NULL</code> .                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>anchor.coordinates</code> | Optional numeric coordinate matrix of off-sample local model anchors with the same number of columns as <code>X</code> . This option is currently supported only with coordinate support distances, not with <code>support.metric = "graph.geodesic"</code> .                                                                                                                                                                                                                                                                                 |
| <code>degree</code>             | Local polynomial degree, one of <code>0L</code> , <code>1L</code> , or <code>2L</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>degree.grid</code>        | Optional degree grid used when <code>support.selection</code> is <code>"cv"</code> or <code>"gcv"</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>support.type</code>       | Support construction rule. <code>"knn"</code> uses exactly <code>support.size</code> nearest observations including the anchor itself for full-data local fitting supports. Prediction supports are then rebuilt from the induced local radius, so tied targets at the kNN boundary can be covered even when they were not part of the exact-size fitting support. <code>"adaptive.radius"</code> uses a radius large enough to include at least <code>min.support</code> observations. <code>"fixed.radius"</code> uses the supplied radius. |
| <code>support.size</code>       | Number of nearest observations for <code>support.type = "knn"</code> . If <code>NULL</code> , defaults to <code>min.support</code> .                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>support.grid</code>       | Optional kNN support-size grid used when <code>support.selection</code> is <code>"cv"</code> or <code>"gcv"</code> .                                                                                                                                                                                                                                                                                                                                                                                                                          |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| radius                    | Radius for support.type = "fixed.radius" and optional base radius for support.type = "adaptive.radius".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| radius.grid               | Optional radius grid used when support.selection is "cv" or "gcv".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| min.support               | Minimum positive-weight support size. If NULL, defaults to the local polynomial design size plus support.buffer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| min.support.grid          | Optional minimum-support grid used when support.selection is "cv" or "gcv".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| support.buffer            | Nonnegative integer added to the local design size when deriving the default min.support.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| kernel                    | Kernel used for local fitting and model averaging.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| kernel.grid               | Optional kernel grid used when support.selection is "cv" or "gcv".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| model.weight.rule         | Optional per-anchor model-quality multiplier applied to the ordinary kernel averaging weights. "none" preserves the kernel-only averaging from earlier phases. "condition" downweights poorly conditioned local designs with $q_u^{\text{cond}} = 1/\max(\kappa_u, 1)$ . "support" upweights anchors with larger positive fitting supports using support size divided by the maximum support size. "boundary" downweights asymmetric boundary-like supports with $q_u^{\text{bdry}} = 1/(1+b_u)$ . "quality" multiplies these three component weights. For every non-"none" rule, diagnostics\$model.weight.raw stores the selected unnormalized rule multiplier, while model.weights and diagnostics\$model.weights store the final multipliers after median-one normalization over positive raw values. Raw zero or non-finite multipliers are retained diagnostically but are replaced by a tiny positive final floor before prediction averaging, so model-quality rules reweight existing prediction support rather than removing anchors from support membership. These weights affect prediction averaging only; they do not change local supports or local coefficient estimation. |
| duplicate.action          | How duplicate coordinate rows should be handled. "keep" allows duplicates, records duplicate diagnostics, and preserves observed-anchor self-inclusion. "error" rejects duplicate rows. Jittering duplicates is deliberately not implemented.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| coordinate.method         | Coordinate system used for each local polynomial design. "coordinates" uses supplied coordinates centered at the anchor. "local.pca" uses a deterministic local PCA chart estimated from the anchor support.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| chart.dim                 | Local chart dimension for coordinate.method = "local.pca". If NULL, defaults to ncol(X). The special value "auto" estimates a single observed-data local PCA dimension without using responses, truth values, latent coordinates, or labels. For coordinate.method = "coordinates", chart.dim must be NULL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| support.metric            | Distance system used for support construction. "coordinates" uses Euclidean distances in X. "graph.geodesic" uses weighted shortest-path distances from graph or adj.list/weight.list. "auto" uses graph geodesic distances when graph input is supplied and coordinate distances otherwise.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| auto.chart.support.metric | Support system used when chart.dim = "auto". "coordinates" uses Euclidean coordinate neighborhoods, "operator" uses the resolved MALPS support metric, and "both" computes both diagnostics side by side.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

|                                             |                                                                                                                                                                                                                                                                                                               |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>auto.chart.selection.metric</code>    | Which auto chart-dimension diagnostic to use for the fitted MALPS model when both diagnostics are available. The default "coordinates" preserves historical behavior.                                                                                                                                         |
| <code>support.selection</code>              | "fixed" for a single support profile, "cv" to tune support parameters by cross-validation, or "gcv" to tune support parameters by exact dense generalized cross-validation. The GCV path is cached across candidates and is exact for fixed-weight linear MALPS fits; it rejects robust residual reweighting. |
| <code>foldid</code>                         | Optional integer fold assignments. If supplied, <code>cv.repeats</code> is forced to one.                                                                                                                                                                                                                     |
| <code>cv.folds</code>                       | Number of folds generated when <code>foldid = NULL</code> .                                                                                                                                                                                                                                                   |
| <code>cv.loss</code>                        | Cross-validation loss.                                                                                                                                                                                                                                                                                        |
| <code>cv.repeats</code>                     | Number of independent fold assignments generated when <code>foldid = NULL</code> .                                                                                                                                                                                                                            |
| <code>cv.seed</code>                        | Optional seed for generated folds.                                                                                                                                                                                                                                                                            |
| <code>cv.one.se</code>                      | Logical; use a one-standard-error rule for support selection.                                                                                                                                                                                                                                                 |
| <code>gcv.exact.max.n</code>                | Maximum number of observations allowed for exact dense GCV support selection. Larger datasets should use "cv" until a sparse or trace-estimated GCV path is implemented.                                                                                                                                      |
| <code>local.solver</code>                   | Local weighted least-squares solver. "auto" uses normal equations only for full-rank, well-conditioned local designs and otherwise falls back to SVD.                                                                                                                                                         |
| <code>normal.equations.max.condition</code> | Maximum local design condition number for normal equations under <code>local.solver = "auto"</code> .                                                                                                                                                                                                         |
| <code>robust.iterations</code>              | Number of Cleveland-style robust residual reweighting iterations applied inside each local polynomial fit. The default 0L preserves ordinary MALPS. Positive values multiply the geometric fitting weights by Tukey bisquare residual weights recomputed from the local fit.                                  |
| <code>robust.tuning.constant</code>         | Positive bisquare tuning constant. The default 6 matches the usual LOWESS convention in which residuals are scaled by $6 * \text{median}(\text{abs}(\text{residuals}))$ .                                                                                                                                     |
| <code>verbose</code>                        | Logical; reserved for future progress messages.                                                                                                                                                                                                                                                               |
| <code>...</code>                            | Reserved for future extensions. Supplying unused arguments is an error.                                                                                                                                                                                                                                       |

## Details

Current development benchmarks suggest using the plain MALPS defaults for production-style fits unless a specific diagnostic motivates a non-default option: `support.selection = "cv"`, `robust.iterations = 0L`, `model.weight.rule = "none"`, and observed anchors (`anchor.coordinates = NULL`). Exact GCV selection (`support.selection = "gcv"`) is a useful faster option for fixed-weight, non-robust MALPS fits, especially exploratory continuous smoothing runs, but it is not yet the universal default. Binary responses are treated as numeric conditional-expectation targets; fitted values are not clipped inside the smoother. Grid anchors, empirical-Bayes shrinkage, robust local residual reweighting, and conservative/smoothed GCV selection remain experimental controls rather than default recommendations.

Graph construction is supplied by **dgraphs** or by explicit `adj.list/weight.list` payloads. Coordinate MALPS paths and graph-geodesic payload validation are package-local; shortest-path distances are computed through **dgraphs**.

### Value

A list of class "malps" with fitted values, local model coefficients, supports, averaging weights, diagnostics, and selection metadata.

### Choosing controls

**First fit:** supply `X`, `y`, `degree`, and `support.type` with a suitable `support.size` or `radius`. Coordinate-distance fits support new predictions.

**Selection:** `support` and `degree` grids, `foldid`, and `cv.folds` govern held-out selection. Robust refits recompute residual weights; selection is not automatically repeated by `refit`.

**Geometry:** `support.metric`, graph inputs, anchors, and chart controls determine the local models. Graph-geodesic fits require aligned vertices and do not currently predict at new targets.

**Numerics:** `local.solver`, conditioning controls, and robust weighting have distinct meanings. Keep their defaults until diagnostics identify a need.

**Diagnostics:** inspect selected, model weights, and stored local fits. `smoother.matrix` allocates a dense matrix and applies only to fixed weighting; its `max.n` guard makes that cost explicit.

### Examples

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.malps(X, sin(2 * pi * X[, 1]), degree = 1L,
               support.type = "knn", support.size = 8L)
head(fit$fitted.values)
```

---

fit.masked.graph.field

*Fit a Graph Field from Partially Observed Responses*

---

### Description

Estimate a conditional mean by weighted least squares with an unnormalized graph Laplacian penalty. Repeated observations at a vertex remain separate. Unobserved components return NA rather than an arbitrary constant.

### Usage

```
fit.masked.graph.field(operator, vertex, y, lambda, weights = NULL)
```

**Arguments**

|          |                                                                    |
|----------|--------------------------------------------------------------------|
| operator | An unnormalized metric.graph.lowpass.operator.                     |
| vertex   | Integer vertex index for each observed response.                   |
| y        | Finite numeric observed responses (no missing-value placeholders). |
| lambda   | Positive smoothing parameter.                                      |
| weights  | Positive observation weights; normalized to sum to one.            |

**Value**

A list with vertex fitted values, support flags, observation weights, objective, and relative linear-system residual.

---

fit.metric.graph.lowpass

*Fit Metric-Conductance Graph Low-Pass Regression*


---

**Description**

Fits graph-spectral low-pass regression on a supplied graph by transforming metric edge lengths into conductances and smoothing the response in the eigenbasis of the weighted graph Laplacian.

**Usage**

```
fit.metric.graph.lowpass(
  adj.list,
  weight.list,
  y,
  conductance.rule = c("inverse.length.power", "exp.length", "exp.length.squared",
    "self.tuned.gaussian"),
  conductance.epsilon = 1e-08,
  conductance.alpha = 1,
  conductance.sigma = NULL,
  conductance.sigma.rule = c("edge.quantile", "median", "local.k"),
  conductance.sigma.quantile = 0.75,
  conductance.local.k = 5L,
  laplacian.type = c("unnormalized", "symmetric.normalized"),
  n.eigenpairs = 50L,
  filter.type = c("heat_kernel", "tikhonov", "cubic_spline", "gaussian", "exponential",
    "butterworth"),
  eta.grid = NULL,
  n.candidates = 40L,
  eigen.solver = c("auto", "sparse", "dense"),
  dense.eigen.threshold = 200L,
  dense.fallback.threshold = 5000L,
  dense.fallback = c("auto", "never", "always"),
```

```

verbose = FALSE,
eta.search = c("fixed", "guarded.gcv"),
eta.expansion.factor = 3,
eta.max.expansions = 3L,
eta.identity.departure = 0.01,
eta.truncation.tol = 1e-04
)

```

## Arguments

|                                         |                                                                                                                                                                                                                                                                          |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list</code>                   | List of integer neighbor vectors using 1-based vertex indices.                                                                                                                                                                                                           |
| <code>weight.list</code>                | List of positive metric edge lengths parallel to <code>adj.list</code> . These are interpreted as edge lengths, not conductances.                                                                                                                                        |
| <code>y</code>                          | Numeric response vector of length <code>length(adj.list)</code> .                                                                                                                                                                                                        |
| <code>conductance.rule</code>           | Character scalar. One of "inverse.length.power", "exp.length", "exp.length.squared", or "self.tuned.gaussian".                                                                                                                                                           |
| <code>conductance.epsilon</code>        | Positive numeric regularizer used by inverse-power conductances and as a local-scale floor.                                                                                                                                                                              |
| <code>conductance.alpha</code>          | Positive numeric exponent for "inverse.length.power".                                                                                                                                                                                                                    |
| <code>conductance.sigma</code>          | Optional positive global scale for exponential rules. If NULL, it is selected by <code>conductance.sigma.rule</code> .                                                                                                                                                   |
| <code>conductance.sigma.rule</code>     | Rule for selecting a global scale when needed.                                                                                                                                                                                                                           |
| <code>conductance.sigma.quantile</code> | Quantile used when <code>conductance.sigma.rule = "edge.quantile"</code> .                                                                                                                                                                                               |
| <code>conductance.local.k</code>        | Positive integer local incident-edge order statistic for "self.tuned.gaussian".                                                                                                                                                                                          |
| <code>laplacian.type</code>             | Laplacian operator. "unnormalized" uses the weighted graph Laplacian $L = D - C$ . "symmetric.normalized" uses $L_{\text{sym}} = I - D^{-1/2}CD^{-1/2}$ .                                                                                                                |
| <code>n.eigenpairs</code>               | Positive integer number of eigenpairs to compute.                                                                                                                                                                                                                        |
| <code>filter.type</code>                | Spectral low-pass filter family.                                                                                                                                                                                                                                         |
| <code>eta.grid</code>                   | Optional numeric eta grid. Values must be positive except for <code>filter.type = "heat_kernel"</code> , where <code>eta = 0</code> is allowed and gives the identity/no-smoothing limit. If NULL, the existing package helper <code>generate.eta.grid()</code> is used. |
| <code>n.candidates</code>               | Number of eta candidates when <code>eta.grid = NULL</code> .                                                                                                                                                                                                             |
| <code>eigen.solver</code>               | "auto", "sparse", or "dense". "auto" uses dense decomposition only for $n \leq \text{dense.eigen.threshold}$ , then sparse-first.                                                                                                                                        |
| <code>dense.eigen.threshold</code>      | Exact dense threshold for auto mode. Default 200L, intended for small reference/testing problems.                                                                                                                                                                        |

|                          |                                                                                                                                                                                                                                           |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dense.fallback.threshold | Maximum graph size for emergency dense fallback when sparse decomposition fails and fallback is allowed.                                                                                                                                  |
| dense.fallback           | "auto", "never", or "always".                                                                                                                                                                                                             |
| verbose                  | Logical. Reserved for future diagnostic messages.                                                                                                                                                                                         |
| eta.search               | Heat-time search controller. "fixed" evaluates the supplied or generated grid once. "guarded.gcv" is available only for filter.type = "heat_kernel" and proposes one lower positive time whenever GCV selects the current lower endpoint. |
| eta.expansion.factor     | Geometric divisor used by guarded GCV lower-time expansion.                                                                                                                                                                               |
| eta.max.expansions       | Maximum number of guarded GCV lower-time extensions.                                                                                                                                                                                      |
| eta.identity.departure   | Smallest allowed departure from the identity at the largest retained graph-Laplacian eigenvalue.                                                                                                                                          |
| eta.truncation.tol       | Omitted-mode attenuation tolerance used to certify guarded proposals for a truncated basis.                                                                                                                                               |

**Value**

A list of class "metric.graph.lowpass.fit".

**References**

Gajer, P. and Ravel, J. (2025). Adaptive geometric regression for high-dimensional structured data. arXiv:2511.03817. doi:10.48550/arXiv.2511.03817

**Examples**

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
lengths <- list(1, c(1, 1), c(1, 1), 1)
fit.metric.graph.lowpass(
  adj, lengths, y = c(0, 0.2, 1.8, 2), n.eigenpairs = 4L,
  eta.grid = c(0.1, 1), eigen.solver = "dense"
)
```

**Description**

Experimental R reference implementation of prediction-synchronized local polynomial smoothing (PS-LPS). For fixed local polynomial parameters, PS-LPS fits one local polynomial chart per anchor and adds a quadratic penalty that synchronizes chart predictions on overlap points.

**Usage**

```

fit.ps.lps(
  X,
  y,
  foldid = NULL,
  support.size = NULL,
  degree = 2L,
  kernel = "gaussian",
  chart.dim = NULL,
  support.grid = NULL,
  degree.grid = NULL,
  kernel.grid = NULL,
  auto.chart.support.metric = c("coordinates", "operator", "both"),
  auto.chart.selection.metric = c("coordinates", "operator"),
  chart.dim.grid = NULL,
  selection.strategy = c("grid", "sparse_kd", "plateau_kd"),
  chart.dim.max = NULL,
  design.margin = 2L,
  lambda.sync.grid = c(0, 0.001, 0.01, 0.1, 1, 10),
  lambda.sync.search = c("grid", "guarded"),
  lambda.sync.selection = c("cv", "fixed"),
  local.candidate.search = c("screened", "full", "subgrid"),
  local.candidate.search.control = list(),
  lambda.sync.search.control = list(),
  lambda.diagnostics = c("all", "selected"),
  lambda.ridge = 1e-08,
  design.basis = c("monomial", "weighted.qr", "weighted.qr.drop",
    "orthogonal.polynomial.drop"),
  design.drop.tol = sqrt(.Machine$double.eps),
  ridge.multiplier.grid = NULL,
  ridge.condition.max = Inf,
  sync.neighbor.size = NULL,
  overlap.weight = c("normalized.product", "product"),
  chart.activation = c("none", "subject.od"),
  chart.activation.response = NULL,
  chart.activation.control = list(),
  ps.lps.geometry.cache = NULL,
  ps.lps.local.pca.supports = NULL,
  cv.folds = 5L,
  cv.seed = 1L
)

```

**Arguments**

|        |                                                    |
|--------|----------------------------------------------------|
| X      | Numeric covariate matrix.                          |
| y      | Numeric response vector.                           |
| foldid | Optional integer cross-validation fold assignment. |

|                                          |                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>support.size</code>                | Optional single neighborhood size for the fixed local setup path. Use <code>support.grid</code> for CV selection over neighborhood sizes.                                                                                                                                                                                                                                                                       |
| <code>degree</code>                      | Single local polynomial degree for the fixed local setup path. Use <code>degree.grid</code> for CV selection over degrees.                                                                                                                                                                                                                                                                                      |
| <code>kernel</code>                      | Single kernel name for the fixed local setup path. Use <code>kernel.grid</code> for CV selection over kernels.                                                                                                                                                                                                                                                                                                  |
| <code>chart.dim</code>                   | Chart dimension for the local PCA charts. A scalar fixes one dimension for all anchors; an integer vector supplies one local chart dimension per anchor. The special value "auto" estimates one global chart dimension from observed $X$ only. The experimental special value "local.auto" estimates one local chart dimension per anchor, using the same local-PCA dimension rule as <a href="#">fit.lps</a> . |
| <code>support.grid</code>                | Candidate neighborhood sizes. When supplied, PS-LPS selects over support size, degree, kernel, and synchronization strength by materialized-fold CV. If both <code>support.size</code> and <code>support.grid</code> are absent, defaults to <code>c(10L, 15L, 20L)</code> , matching <a href="#">fit.lps</a> .                                                                                                 |
| <code>degree.grid</code>                 | Candidate local polynomial degrees for grid selection. Defaults to the scalar degree.                                                                                                                                                                                                                                                                                                                           |
| <code>kernel.grid</code>                 | Candidate kernels for grid selection. Defaults to the scalar kernel.                                                                                                                                                                                                                                                                                                                                            |
| <code>auto.chart.support.metric</code>   | Support system used when <code>chart.dim = "auto"</code> or <code>"local.auto"</code> . Because PS-LPS uses coordinate supports, "operator" is equivalent to "coordinates".                                                                                                                                                                                                                                     |
| <code>auto.chart.selection.metric</code> | Which auto chart-dimension diagnostic to select when both diagnostics are requested.                                                                                                                                                                                                                                                                                                                            |
| <code>chart.dim.grid</code>              | Optional candidate chart dimensions for experimental local-candidate selection. Numeric grids can be evaluated with <code>selection.strategy = "sparse_kd"</code> .                                                                                                                                                                                                                                             |
| <code>selection.strategy</code>          | Candidate-selection strategy. "grid" evaluates the requested candidate grid. "sparse_kd" evaluates a sparse support-size by chart-dimension skeleton when <code>chart.dim.grid</code> is supplied. "plateau_kd" uses the same geometry-only support-size and chart-dimension plateau rule as <a href="#">fit.lps</a> .                                                                                          |
| <code>chart.dim.max</code>               | Optional explicit maximum chart dimension for the sparse coupled candidate family.                                                                                                                                                                                                                                                                                                                              |
| <code>design.margin</code>               | Nonnegative integer feasibility margin used to screen local polynomial design size before candidate evaluation.                                                                                                                                                                                                                                                                                                 |
| <code>lambda.sync.grid</code>            | Candidate synchronization strengths.                                                                                                                                                                                                                                                                                                                                                                            |
| <code>lambda.sync.search</code>          | Lambda-search policy. "grid" evaluates the supplied grid exactly. "guarded" uses an experimental guarded coarse-to-refine search with boundary expansion.                                                                                                                                                                                                                                                       |
| <code>lambda.sync.selection</code>       | Lambda-selection mode. "cv" performs the usual internal fold-weighted lambda selection. "fixed" treats the single value in <code>lambda.sync.grid</code> as preselected and skips internal lambda CV; this is intended for outer selection loops that already score scalar candidates.                                                                                                                          |

`local.candidate.search`

Local-candidate search policy when `support.grid`, `degree.grid`, or `kernel.grid` contains more than one candidate. "screened" is the routine default: it first ranks candidates by ordinary LPS materialized-fold CV, then runs PS-LPS only on a screened subset plus guard candidates. "full" evaluates PS-LPS lambda search for every local candidate and is the exact audit/reference path. "subgrid" skips the ordinary-LPS screening pass and evaluates only a deterministic support/kernel guard subgrid; this is intended for high-dimensional preflight runs where the screening pass is itself too expensive.

`local.candidate.search.control`

Optional list controlling screened local-candidate search. Supported fields are `top.n` (default 8), `max.candidates` (default 12), `neighbor.radius` (default 1), and `guard.support.quantiles` (default `c(0, 0.5, 1)`).

`lambda.sync.search.control`

Optional list controlling guarded search. Supported fields are `coarse.size` (default 5), `refine.radius` (default 2), `rel.tol` (default 0.002), `boundary.guard.rel.tol` (default 0.01), `boundary.expand` (default TRUE), `boundary.factor` (default 3), `max.boundary.expansions` (default 2), and `max.candidates` (default 25). Boundary expansion may evaluate positive candidates outside the supplied `lambda.sync.grid`. `max.candidates` is a global cap on distinct evaluated candidates; a very small cap can prevent local refinement or boundary expansion.

`lambda.diagnostics`

Lambda-diagnostic evaluation policy. "all" computes synchronization-energy and local-GCV diagnostics for every evaluated lambda and is the audit/reference mode. "selected" computes those diagnostics only for the selected lambda, which is faster for routine selection.

`lambda.ridge`

Nonnegative scale-relative ridge used in the chart coefficient solve. Use 0 for the unregularized least-squares model.

`design.basis`

Local polynomial design backend. See [fit.lps](#). In PS-LPS, "weighted.qr.drop" drops numerically dependent columns separately in each anchor chart before the synchronized system is assembled, and "orthogonal.polynomial.drop" builds each synchronized chart in a weighted-orthogonal polynomial basis.

`design.drop.tol`

Relative QR tolerance used by `design.basis = "weighted.qr.drop"` or `design.basis = "orthogonal.polynomial.drop"`.

`ridge.multiplier.grid`

Optional nonnegative ridge multipliers for adaptive scale-relative ridge selection. When supplied, the solver uses the smallest multiplier whose penalized normal equations pass `ridge.condition.max`. If NULL, `lambda.ridge` is used as the single multiplier for backward compatibility.

`ridge.condition.max`

Maximum allowed condition number for adaptive ridge selection. Use Inf to disable the condition-number guard.

`sync.neighbor.size`

Number of nearby anchor pairs considered for synchronization from each anchor support.

`overlap.weight` Overlap weighting rule.

|                           |                                                                                                                                                                                                                                                                                    |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| chart.activation          | Optional sparse-response chart activation rule. "none" preserves ordinary PS-LPS. "subject.od" is intended for subject-occupation density workflows and deactivates local charts with no subject mass, too few positive subject-visited support points, or only fringe occupation. |
| chart.activation.response | Optional nonnegative vector used only for chart.activation = "subject.od" to decide whether each chart is active. When omitted, y is used.                                                                                                                                         |
| chart.activation.control  | List controlling sparse chart activation; see <a href="#">fit.lps</a> .                                                                                                                                                                                                            |
| ps.lps.geometry.cache     | Optional precomputed geometry cache used by occupation-density cross-validation. This is an implementation detail; ordinary callers should leave it as NULL.                                                                                                                       |
| ps.lps.local.pca.supports | Optional precomputed local-PCA supports used when activation-specific frames must be rebuilt. This is an implementation detail; ordinary callers should leave it as NULL.                                                                                                          |
| cv.folds                  | Number of folds when foldid is absent.                                                                                                                                                                                                                                             |
| cv.seed                   | Fold seed when foldid is absent.                                                                                                                                                                                                                                                   |

**Value**

A list with fitted values, selected lambda, CV table, diagnostics, and fitted chart coefficients.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit.ps.lps(
  X, sin(2 * pi * X[, 1]), support.size = 7L, degree = 1L,
  chart.dim = 1L, lambda.sync.grid = 0,
  lambda.sync.selection = "fixed", cv.folds = 2L
)
```

---

fit.pttf.trend.filtering

*Fit A Parallel-Transport Trend Filter From A PTTF Operator*

---

**Description**

Fits experimental PTTF regression models from a transported difference operator produced by [pttf.operator](#). The implementation focuses on one response vector and explicit operator-row policies, especially the 1-D boundary policy used to separate interior transported rows from end-point rows.

**Usage**

```

fit.pttf.trend.filtering(
  geometry = NULL,
  operator = NULL,
  X = NULL,
  y,
  derivative.order = 3L,
  penalty = c("l1", "l2"),
  lambda.grid = NULL,
  lambda.selection = c("cv", "fixed"),
  n.lambda = 80L,
  nfolds = 5L,
  foldid = NULL,
  cv.loss = c("mse", "mae"),
  selection = c("min", "one.se"),
  lambda.extension = c("none", "auto"),
  solver = c("genlasso", "admm", "auto"),
  operator.row.policy = c("all", "drop.line.boundary", "diagnostic.only"),
  line.order = NULL,
  boundary.trim = NULL,
  row.mass.rule = c("none", "node.mass"),
  row.normalize = c("none", "l2"),
  weights = NULL,
  maxsteps = 2000L,
  minlam = 0,
  approx = FALSE,
  rtol = 1e-07,
  btol = 1e-07,
  eps = 1e-04,
  admm.rho = NULL,
  admm.maxiter = 2000L,
  admm.abstol = 1e-04,
  admm.reltol = 0.001,
  verbose = FALSE,
  admm.adaptive.rho = TRUE,
  ...
)

```

**Arguments**

|                       |                                                                                                                                                   |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>geometry</code> | Optional "pttf_geometry" object. Used only when operator is NULL.                                                                                 |
| <code>operator</code> | Optional "pttf_operator" object. Supplying an operator is recommended for validation so geometry/operator construction is separated from fitting. |
| <code>X</code>        | Optional data matrix used to build <code>pttf.geometry</code> when both geometry and operator are NULL.                                           |
| <code>y</code>        | Numeric response vector.                                                                                                                          |

|                                                  |                                                                                                                                                                                                                                                                                         |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| derivative.order                                 | Integer derivative order passed to <a href="#">pttf.operator</a> when an operator must be built.                                                                                                                                                                                        |
| penalty                                          | "l1" for generalized-lasso style fitting or "l2" for the quadratic energy fit.                                                                                                                                                                                                          |
| lambda.grid                                      | Optional lambda grid. Required for <code>lambda.selection = "fixed"</code> .                                                                                                                                                                                                            |
| lambda.selection                                 | "cv" or "fixed".                                                                                                                                                                                                                                                                        |
| n.lambda                                         | Number of lambdas in a generated grid.                                                                                                                                                                                                                                                  |
| nfolds                                           | Number of folds used only when <code>foldid = NULL</code> .                                                                                                                                                                                                                             |
| foldid                                           | Optional integer fold assignments. Validation lanes should pass a materialized <code>foldid</code> ; otherwise folds are generated by R's current RNG state.                                                                                                                            |
| cv.loss                                          | Cross-validation loss, "mse" or "mae".                                                                                                                                                                                                                                                  |
| selection                                        | Lambda selection rule, "min" or "one.se".                                                                                                                                                                                                                                               |
| lambda.extension                                 | Currently recorded as metadata; no extension heuristic is implemented.                                                                                                                                                                                                                  |
| solver                                           | L1 solver backend.                                                                                                                                                                                                                                                                      |
| operator.row.policy                              | Which operator rows to use for fitting.                                                                                                                                                                                                                                                 |
| line.order                                       | Vertex order for 1-D boundary-row policies.                                                                                                                                                                                                                                             |
| boundary.trim                                    | Number of endpoint ranks to drop for <code>operator.row.policy = "drop.line.boundary"</code> . Defaults to <code>derivative.order</code> .                                                                                                                                              |
| row.mass.rule, row.normalize                     | Passed to <a href="#">pttf.operator</a> when this function builds an operator. <code>row.normalize</code> is also used as the L1 solver row scaling for the supplied fit operator.                                                                                                      |
| weights                                          | Optional nonnegative observation weights.                                                                                                                                                                                                                                               |
| maxsteps, minlam, approx, rtol, btol, eps        | Genlasso controls for L1 fits.                                                                                                                                                                                                                                                          |
| admm.rho, admm.maxiter, admm.abstol, admm.reltol | ADMM controls used when the L1 backend is "admm" or falls back to ADMM. A NULL initial penalty is chosen from the operator scale. The shared solver's convergence checks, failure behavior, and CV eligibility rules are described in <a href="#">fit.ssrhe.hessian.l1.regression</a> . |
| verbose                                          | Logical.                                                                                                                                                                                                                                                                                |
| admm.adaptive.rho                                | Logical; adapt the ADMM penalty during the initial iterations. Set FALSE to keep its initial value fixed.                                                                                                                                                                               |
| ...                                              | Additional arguments passed to <a href="#">pttf.geometry</a> or <a href="#">pttf.operator</a> when those objects are built internally.                                                                                                                                                  |

**Value**

A list of class "pttf.trend.filtering.fit".

## Examples

```
n <- 10L
X <- matrix(seq(0, 1, length.out = n), ncol = 1)
adj <- lapply(seq_len(n), function(i) intersect(c(i - 1L, i + 1L), 1:n))
lengths <- Map(function(i, j) abs(X[j, 1] - X[i, 1]), seq_len(n), adj)
geometry <- pttf.geometry(
  X, adj, lengths, graph = "supplied", tangent.dim = 1L
)
operator <- pttf.operator(geometry, derivative.order = 2L)
fit.pttf.trend.filtering(
  operator = operator, y = sin(2 * pi * X[, 1]), penalty = "l2",
  lambda.grid = 0.1, lambda.selection = "fixed"
)
```

---

fit.slpl.tf

*Fit A Fixed-Operator Synchronized LPL-TF Model*


---

## Description

Fits the fixed-operator S-LPL-TF estimator

$$\hat{f} = \arg \min_f \frac{1}{2} \|y - f\|_2^2 + \lambda_1 \|A_{\text{LPL}} f\|_1 + \frac{\lambda_2}{2} \|C_{\text{sync}} f\|_2^2.$$

## Usage

```
fit.slpl.tf(
  X = NULL,
  y,
  operator = NULL,
  lambda1 = NULL,
  lambda2 = 0,
  lambda1.grid = NULL,
  lambda2.grid = NULL,
  lambda.selection = c("fixed", "cv"),
  operator.grid = NULL,
  foldid = NULL,
  cv.folds = 5L,
  cv.loss = c("mse", "rmse", "mae"),
  cv.seed = NULL,
  cv.repeats = 1L,
  solver = c("genlasso"),
  selection = c("min", "one.se"),
  maxsteps = 2000L,
  minlam = 0,
  approx = FALSE,
  rtol = 1e-07,
  btol = 1e-07,
```

```

    eps = 1e-04,
    verbose = FALSE,
    ...
  )

```

## Arguments

|                                                        |                                                                                                                                                                                                              |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                                         | Optional coordinate matrix used to build <code>slpl.tf.operator</code> when operator is NULL.                                                                                                                |
| <code>y</code>                                         | Numeric response vector.                                                                                                                                                                                     |
| <code>operator</code>                                  | Optional "slpl_tf_operator" object.                                                                                                                                                                          |
| <code>lambda1</code>                                   | Fixed LPL-TF $\ell_1$ penalty, or a shortcut for <code>lambda1.grid</code> when <code>lambda.selection = "cv"</code> .                                                                                       |
| <code>lambda2</code>                                   | Fixed quadratic synchronization penalty, or a shortcut for <code>lambda2.grid</code> when <code>lambda.selection = "cv"</code> .                                                                             |
| <code>lambda1.grid</code>                              | Optional nonnegative $\lambda_1$ grid for CV.                                                                                                                                                                |
| <code>lambda2.grid</code>                              | Optional nonnegative $\lambda_2$ grid for CV.                                                                                                                                                                |
| <code>lambda.selection</code>                          | "fixed" or "cv". CV uses materialized folds and a deterministic Cartesian grid over <code>lambda1.grid</code> and <code>lambda2.grid</code> .                                                                |
| <code>operator.grid</code>                             | Optional operator candidate grid. Supply a data frame with one row per candidate or a list of named lists.                                                                                                   |
| <code>foldid</code>                                    | Optional deterministic fold assignments for CV.                                                                                                                                                              |
| <code>cv.folds</code>                                  | Number of generated folds when <code>foldid = NULL</code> .                                                                                                                                                  |
| <code>cv.loss</code>                                   | Cross-validation loss, "mse", "rmse", or "mae".                                                                                                                                                              |
| <code>cv.seed</code>                                   | Optional seed for reproducible generated folds.                                                                                                                                                              |
| <code>cv.repeats</code>                                | Number of CV repeats. The current implementation supports only one repeat.                                                                                                                                   |
| <code>solver</code>                                    | Current implementation supports only "genlasso".                                                                                                                                                             |
| <code>selection</code>                                 | Selection rule, "min" or "one.se". For the two-parameter one-standard-error rule, the most regularized eligible pair is chosen by decreasing <code>lambda1</code> and then decreasing <code>lambda2</code> . |
| <code>maxsteps, minlam, approx, rtol, btol, eps</code> | Genlasso controls.                                                                                                                                                                                           |
| <code>verbose</code>                                   | Logical.                                                                                                                                                                                                     |
| <code>...</code>                                       | Additional arguments passed to <code>slpl.tf.operator</code> when the operator is built internally.                                                                                                          |

## Value

A list of class "slpl\_tf".

**Examples**

```

if (requireNamespace("genlasso", quietly = TRUE)) {
  X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
  fit.spl.tf(X, sin(2 * pi * X[, 1]), degree = 1L,
            support.type = "knn", support.size = 7L,
            lambda1 = 0.1, lambda2 = 0, lambda.selection = "fixed")
}

```

---

```
fit.ssrhe.hessian.l1.regression
```

*Fit SSRHE-Style Hessian L1 Regression*

---

**Description**

Fits an  $\ell_1$ -adaptive SSRHE Hessian comparator using the row operator  $A$  from [ssrhe.hessian.operator](#):

$$\hat{f}_\lambda = \arg \min_f \left\{ \frac{1}{2} \sum_i w_i (y_i - f_i)^2 + \lambda \|Af\|_1 \right\}.$$

**Usage**

```

fit.ssrhe.hessian.l1.regression(
  X,
  y,
  k = NULL,
  tangent.dim,
  lambda.grid = NULL,
  lambda.selection = c("cv", "fixed"),
  weights = NULL,
  n.lambda = 40L,
  nfolds = 5L,
  fold.id = NULL,
  loss = c("mse", "mae"),
  selection = c("min", "one.se"),
  nn.index = NULL,
  neighborhood.type = c("knn", "adaptive.radius", "supplied"),
  support.index = NULL,
  adaptive.k.scale = NULL,
  radius.rule = c("geomean", "max", "min"),
  radius.factor = 1.25,
  min.support = NULL,
  max.support = NULL,
  support.buffer = 2L,
  support.topup = c("nearest", "none"),
  tangent.dim.rule = c("fixed", "eigen.cumulative"),
  eigen.tolerance = 0.95,
  derivative.order = 2L,

```

```

pinv.tol = sqrt(.Machine$double.eps),
local.solver = c("auto", "normal.equations", "svd", "qr"),
normal.equations.max.condition = 10000,
solver = c("genlasso", "admm", "auto"),
row.scaling = c("none", "l2"),
admm.rho = NULL,
admm.maxiter = 2000L,
admm.abstol = 1e-06,
admm.reltol = 1e-05,
maxsteps = 2000L,
minlam = 0,
approx = FALSE,
rtol = 1e-07,
btol = 1e-07,
eps = 1e-04,
support.selection = c("rule", "cv"),
support.grid = NULL,
support.cv.max.candidates = 8L,
return.local.diagnostics = FALSE,
return.timing = FALSE,
verbose = FALSE,
admm.adaptive.rho = TRUE
)

```

### Arguments

|                               |                                                                                                                                                                                                                                                                                          |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                | Numeric matrix with one observation per row. Distances for the default neighborhood search are Euclidean distances in these coordinates.                                                                                                                                                 |
| <code>y</code>                | Numeric response vector of length <code>nrow(X)</code> . Matrix responses are not yet supported for the $\ell_1$ path.                                                                                                                                                                   |
| <code>k</code>                | Integer number of nearest neighbors per point, including the point itself, used when <code>neighborhood.type = "knn"</code> . This follows the SSRHE Matlab convention. For adaptive-radius neighborhoods, <code>k</code> may be omitted when <code>adaptive.k.scale</code> is supplied. |
| <code>tangent.dim</code>      | Integer tangent dimension used for the local PCA chart. Required when <code>tangent.dim.rule = "fixed"</code> . If <code>NULL</code> and <code>tangent.dim.rule = "eigen.cumulative"</code> , the dimension is selected locally from the PCA variance ratio.                             |
| <code>lambda.grid</code>      | Optional nonnegative lambda grid. For <code>lambda.selection = "fixed"</code> , this must contain exactly one value. For <code>lambda.selection = "cv"</code> , <code>NULL</code> builds a default grid from the fitted full-data generalized-lasso path.                                |
| <code>lambda.selection</code> | "cv" for observed-label K-fold cross-validation or "fixed" for a supplied fixed lambda.                                                                                                                                                                                                  |
| <code>weights</code>          | Optional nonnegative observation weights. Missing responses automatically receive zero weight.                                                                                                                                                                                           |
| <code>n.lambda</code>         | Number of default lambda candidates when <code>lambda.grid = NULL</code> and <code>lambda.selection = "cv"</code> .                                                                                                                                                                      |

|                                         |                                                                                                                                                                                                                                                                            |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>nfolds</code>                     | Number of validation folds over observed positive-weight labels. Ignored when <code>fold.id</code> is supplied.                                                                                                                                                            |
| <code>fold.id</code>                    | Optional integer fold assignments of length <code>nrow(X)</code> .                                                                                                                                                                                                         |
| <code>loss</code>                       | Validation loss, currently "mse" or "mae".                                                                                                                                                                                                                                 |
| <code>selection</code>                  | Selection rule. "min" chooses the smallest mean validation loss. "one.se" chooses the largest lambda within one standard error of the minimum.                                                                                                                             |
| <code>nn.index</code>                   | Optional integer matrix of neighbor indices, with <code>nrow(X)</code> rows and <code>k</code> columns. Each row must contain its center vertex. If NULL, Euclidean k-NN including self is computed in C++.                                                                |
| <code>neighborhood.type</code>          | Local support rule. "knn" uses the original rectangular self-including kNN neighborhoods. "adaptive.radius" builds variable-size supports from <code>dgraphs::create.rknn.graph</code> "supplied" uses <code>support.index</code> directly.                                |
| <code>support.index</code>              | Optional list of integer vectors, one per row of <code>X</code> . Each element gives a variable-size local support and must contain its center vertex.                                                                                                                     |
| <code>adaptive.k.scale</code>           | Integer local-scale <code>k</code> used by <code>dgraphs::create.rknn.graph()</code> when <code>neighborhood.type = "adaptive.radius"</code> .                                                                                                                             |
| <code>radius.rule, radius.factor</code> | Adaptive-radius graph parameters passed to <code>dgraphs::create.rknn.graph()</code> .                                                                                                                                                                                     |
| <code>min.support</code>                | Optional minimum local support size for adaptive-radius supports. Defaults to the local quadratic design size plus <code>support.buffer</code> , clamped to <code>nrow(X)</code> .                                                                                         |
| <code>max.support</code>                | Optional maximum local support size. Oversized adaptive supports are trimmed to the closest <code>max.support</code> vertices while keeping the center vertex.                                                                                                             |
| <code>support.buffer</code>             | Nonnegative integer added to the local quadratic design size when choosing the default <code>min.support</code> .                                                                                                                                                          |
| <code>support.topup</code>              | How undersized adaptive-radius supports are enlarged. "nearest" appends ambient nearest neighbors. "none" leaves supports unchanged and lets the C++ operator reject undersized supports.                                                                                  |
| <code>tangent.dim.rule</code>           | Either "fixed" or "eigen.cumulative".                                                                                                                                                                                                                                      |
| <code>eigen.tolerance</code>            | Cumulative local PCA variance threshold used when <code>tangent.dim.rule = "eigen.cumulative"</code> and <code>tangent.dim = NULL</code> .                                                                                                                                 |
| <code>derivative.order</code>           | Integer derivative order of the local SSRHE-style operator. 2L is the original local Hessian energy. 3L is an experimental third-derivative energy whose rows estimate unique symmetric third-derivative tensor components with factorial and tensor-multiplicity scaling. |
| <code>pinv.tol</code>                   | Nonnegative tolerance multiplier for local pseudoinverses.                                                                                                                                                                                                                 |
| <code>local.solver</code>               | Local least-squares backend used to map local function values to derivative coefficients. "auto" is the default: it uses normal equations when the local design is full rank and has condition number no larger than <code>normal.equations.max.condition</code> ,         |

and otherwise falls back to SVD. "normal.equations" requests the normal-equation solve for full-rank local designs and falls back only on hard numerical failures. "svd" is the most stable reference path. "qr" uses pivoted QR and falls back to SVD for rank-deficient local designs.

normal.equations.max.condition

Positive condition-number guard used by local.solver = "auto" before accepting the normal-equation backend. The default is deliberately conservative because normal-equation solves square the effective condition number and order-3 near-minimum local supports can be numerically fragile.

solver

Solver backend. "genlasso" uses the generalized-lasso path to propose a grid when needed, then refines every candidate with ADMM on the original objective. "admm" skips the path and requires an explicit grid. "auto" also proceeds when path construction fails, provided an explicit grid is supplied.

row.scaling

Optional row scaling for the penalty matrix before solving. "12" scales nonzero rows of  $A$  to unit Euclidean norm.

admm.rho, admm.maxiter, admm.abstol, admm.reltol

ADMM controls used by all backends, including refinement of generalized-lasso candidates. `admm.rho = NULL` chooses an initial algorithm penalty of  $1 / \max(1, \text{mean}(\text{rowSums}(D^2)))$ , where  $D$  is the penalty operator after any requested row scaling. A positive numeric value sets the initial penalty explicitly. This is separate from statistical regularization `lambda.grid`. The iteration limit is a hard cap.

maxsteps, minlam, approx, rtol, btol, eps, verbose

Controls passed to **genlasso**.

support.selection

Support-profile selection rule. "rule" uses the supplied `adaptive.k.scale`, `min.support`, and `max.support`. "cv" is currently supported for `neighborhood.type = "adaptive.radius"` with `lambda.selection = "cv"` and chooses among rows of `support.grid` by an outer response cross-validation loop.

support.grid

Optional data frame of support profiles with columns `adaptive.k.scale`, `min.support`, and optional `max.support`. If `NULL`, `ssrhe.support.grid` builds a compact default grid.

support.cv.max.candidates

Maximum number of support profiles to try when `support.selection = "cv"`.

return.local.diagnostics

Logical. If `TRUE`, compute additional R-side local chart diagnostics such as chart distortion and boundary asymmetry. These diagnostics are useful for operator audits, but can be skipped in fitting and cross-validation paths for speed.

return.timing

Logical. If `TRUE`, attach a phase-level elapsed time table to the returned operator. The timings separate R-side validation, neighborhood/support construction, native local operator construction, optional local diagnostics, sparse matrix assembly, and output finalization. For adaptive-radius neighborhoods, subphase timings are also attached in `neighborhoods$timing`.

admm.adaptive.rho

Logical; adapt the algorithm penalty during the first 1000 iterations by balancing tolerance-normalized primal and dual residuals. Set `FALSE` to hold the initial penalty fixed.

## Details

This function exposes the SSRHE local polynomial derivative estimator as an  $\ell_1$  penalty, rather than as the original SSRHE  $\ell_2$  Hessian-energy penalty used by `fit.ssrhe.hessian.regression`. With `derivative.order = 2L`,  $A$  contains local quadratic/Hessian rows. With `derivative.order = 3L`,  $A$  contains local cubic third-derivative tensor rows with the same symmetric component scaling used by `ssrhe.hessian.operator`. The existing  $\ell_2$  fit solves a linear system involving  $B = A^\top A$ . This function instead keeps the rows of  $A$  and penalizes their absolute values. It is therefore closer in spirit to graph trend filtering, while retaining SSRHE's local-PCA derivative construction.

The default solver backend is **genlasso**. For larger or numerically fragile third-order operators, `solver = "admm"` fits the requested lambda values directly without computing a generalized-lasso path, and all backends use ADMM for the returned coefficients. `solver = "auto"` can additionally recover from path construction errors with an explicit grid. Cross-validation removes held-out labels from the data-fit term by setting their weights to zero, then scores predictions on those held-out labels.

ADMM uses proximal numerical stabilization, which does not add a ridge penalty to the requested objective. Convergence requires primal feasibility, the dual stopping residual, and stationarity of the original weighted L1 objective to meet their reported tolerances. A finite fixed-lambda iterate that reaches its cap is returned with a warning and `solver$converged = FALSE`; it is not a completed fit. Inspect `solver$admm` before using it, and consider a larger iteration limit or an adaptive penalty. Non-finite selected fits are errors.

CV candidates are eligible only when every fold has an acceptable solve. ADMM iterates that do not converge are not scored. The CV result retains `fold.status`, `fold.converged`, `fold.diagnostics`, `n.successful.folds`, and `eligible`. If no candidate qualifies, an error of class `ssrhe_l1_cv_error` retains fold diagnostics for inspection with `tryCatch`. A selected full-data ADMM fit that fails to converge is also an error in CV mode.

A generalized-lasso path alone can be inaccurate for nearly rank-deficient operators. It is retained in path for diagnostics, but its coefficients are not the returned fit. `solver$backend = "genlasso_admm_refined"` identifies this route; `solver$path.objective` records the unrefined objectives at the grid values. Backend warnings, including its numerical ridge for underdetermined designs, are retained in `solver$warnings`. CV uses the original objective without that additional ridge. The default ADMM tolerances are absolute  $1e-6$  and relative  $1e-5$ ; `convergence` describes numerical stopping at those tolerances, not predictive accuracy. With `row.scaling = "l2"`, the reported objective uses the scaled penalty; `energies$penalty.l1` is that penalty norm, while `energies$hessian.l1` retains the unscaled operator norm.

With `support.selection = "cv"`, each adaptive-radius support candidate builds a fresh SSRHE operator and runs the usual lambda CV with shared fold assignments. The selected fit is the candidate with the smallest selected CV error. This is useful when the default support-size rule is too rigid, but it can be much slower than tuning lambda for one fixed operator.

## Value

A list of class `"ssrhe.hessian.l1.fit"` containing fitted values, residuals, selected lambda, the SSRHE operator, generalized-lasso path metadata, lambda-grid fitted values, and CV diagnostics when requested.

### Choosing controls

**First fit:** specify `X`, `y`, `k`, `tangent.dim`, and `lambda.grid`. For a single supplied penalty set `lambda.selection = "fixed"` and `solver = "admm"` to avoid constructing a path.

**Selection:** `fold.id`, `nfolds`, `loss`, and selection control label `CV`. A grid requires one solve per penalty per fold; support `CV` also rebuilds geometry. Incomplete candidates are excluded, not silently scored.

**Geometry:** neighborhood, tangent and derivative controls define `A`. `row.scaling` changes the penalty itself and thus the statistical objective.

**Numerics:** `admm.*` controls stopping and algorithmic penalties for all backends. Path controls (`maxsteps`, `rtol`, `btol`, `eps`) affect the optional proposal path. They do not certify the final original-objective solve.

**Diagnostics:** `solver$admm`, candidate/fold status and objective components describe numerical acceptance. `path` and `beta.grid` can use substantial memory; use a compact explicit grid with ADMM when no full path is needed.

### References

Kim, K. I., Steinke, F., and Hein, M. (2009). Semi-supervised regression using Hessian energy with an application to semi-supervised dimensionality reduction. *Advances in Neural Information Processing Systems 22*. [https://papers.nips.cc/paper\\_files/paper/2009/hash/f4552671f8909587cf485ea990207f3b.html](https://papers.nips.cc/paper_files/paper/2009/hash/f4552671f8909587cf485ea990207f3b.html)

### Examples

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
fit.ssrhe.hessian.l1.regression(
  X, sin(2 * pi * X[, 1]), k = 6L, tangent.dim = 1L,
  lambda.grid = 0.05, lambda.selection = "fixed", solver = "admm"
)
```

---

```
fit.ssrhe.hessian.regression
```

*Fit SSRHE-Style Hessian-Energy Regression*

---

### Description

Fits the  $\ell_2$  Hessian-energy regularized estimator associated with `ssrhe.hessian.operator`. This is a direct SSRHE-style comparator for Hessian smoothing on point clouds; it is not a replacement for graph trend-filtering regression and is distinct from  $\ell_1$ -adaptive graph trend filtering.

### Usage

```
fit.ssrhe.hessian.regression(
  X,
  y,
  k = NULL,
```

```

tangent.dim,
lambda1,
lambda2 = 0,
weights = NULL,
nn.index = NULL,
neighborhood.type = c("knn", "adaptive.radius", "supplied"),
support.index = NULL,
adaptive.k.scale = NULL,
radius.rule = c("geomean", "max", "min"),
radius.factor = 1.25,
min.support = NULL,
max.support = NULL,
support.buffer = 2L,
support.topup = c("nearest", "none"),
tangent.dim.rule = c("fixed", "eigen.cumulative"),
eigen.tolerance = 0.95,
derivative.order = 2L,
stabilizer = lambda2 > 0,
pinv.tol = sqrt(.Machine$double.eps),
local.solver = c("auto", "normal.equations", "svd", "qr"),
normal.equations.max.condition = 10000,
ridge = 0,
return.A = TRUE,
return.local.diagnostics = FALSE,
return.timing = FALSE,
verbose = FALSE,
lambda.selection = c("fixed", "cv", "gcv"),
lambda1.grid = NULL,
lambda2.grid = NULL,
cv.control = list(),
gcv.control = list()
)

```

### Arguments

|                          |                                                                                                                                                                                                                                                                                          |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>           | Numeric matrix with one observation per row. Distances for the default neighborhood search are Euclidean distances in these coordinates.                                                                                                                                                 |
| <code>y</code>           | Numeric response vector of length <code>nrow(X)</code> or numeric matrix with <code>nrow(X)</code> rows. Matrix columns are fit as separate responses. NA values are allowed and are treated as unobserved by setting the corresponding observation weight to zero.                      |
| <code>k</code>           | Integer number of nearest neighbors per point, including the point itself, used when <code>neighborhood.type = "knn"</code> . This follows the SSRHE Matlab convention. For adaptive-radius neighborhoods, <code>k</code> may be omitted when <code>adaptive.k.scale</code> is supplied. |
| <code>tangent.dim</code> | Integer tangent dimension used for the local PCA chart. Required when <code>tangent.dim.rule = "fixed"</code> . If <code>NULL</code> and <code>tangent.dim.rule = "eigen.cumulative"</code> , the dimension is selected locally from the PCA variance ratio.                             |

|                            |                                                                                                                                                                                                                                                                            |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lambda1                    | Nonnegative Hessian-energy penalty multiplier for $f^\top B f = \ A f\ _2^2$ .                                                                                                                                                                                             |
| lambda2                    | Nonnegative supplemental-stabilizer multiplier for $f^\top B_S f$ . If positive, stabilizer must be TRUE. Currently supported only for derivative.order = 2L.                                                                                                              |
| weights                    | Optional nonnegative observation weights. May be NULL, a vector of length nrow(X), or a matrix with the same dimensions as y.                                                                                                                                              |
| nn.index                   | Optional integer matrix of neighbor indices, with nrow(X) rows and k columns. Each row must contain its center vertex. If NULL, Euclidean k-NN including self is computed in C++.                                                                                          |
| neighborhood.type          | Local support rule. "knn" uses the original rectangular self-including kNN neighborhoods. "adaptive.radius" builds variable-size supports from dgraphs::create.rknn.graph "supplied" uses support.index directly.                                                          |
| support.index              | Optional list of integer vectors, one per row of X. Each element gives a variable-size local support and must contain its center vertex.                                                                                                                                   |
| adaptive.k.scale           | Integer local-scale k used by dgraphs::create.rknn.graph() when neighborhood.type = "adaptive.radius".                                                                                                                                                                     |
| radius.rule, radius.factor | Adaptive-radius graph parameters passed to dgraphs::create.rknn.graph().                                                                                                                                                                                                   |
| min.support                | Optional minimum local support size for adaptive-radius supports. Defaults to the local quadratic design size plus support.buffer, clamped to nrow(X).                                                                                                                     |
| max.support                | Optional maximum local support size. Oversized adaptive supports are trimmed to the closest max.support vertices while keeping the center vertex.                                                                                                                          |
| support.buffer             | Nonnegative integer added to the local quadratic design size when choosing the default min.support.                                                                                                                                                                        |
| support.topup              | How undersized adaptive-radius supports are enlarged. "nearest" appends ambient nearest neighbors. "none" leaves supports unchanged and lets the C++ operator reject undersized supports.                                                                                  |
| tangent.dim.rule           | Either "fixed" or "eigen.cumulative".                                                                                                                                                                                                                                      |
| eigen.tolerance            | Cumulative local PCA variance threshold used when tangent.dim.rule = "eigen.cumulative" and tangent.dim = NULL.                                                                                                                                                            |
| derivative.order           | Integer derivative order of the local SSRHE-style operator. 2L is the original local Hessian energy. 3L is an experimental third-derivative energy whose rows estimate unique symmetric third-derivative tensor components with factorial and tensor-multiplicity scaling. |
| stabilizer                 | Logical. If TRUE, also construct the supplemental stabilizer matrix described in the SSRHE supplement. Currently supported only for derivative.order = 2L.                                                                                                                 |
| pinv.tol                   | Nonnegative tolerance multiplier for local pseudoinverses.                                                                                                                                                                                                                 |
| local.solver               | Local least-squares backend used to map local function values to derivative coefficients. "auto" is the default: it uses normal equations when the local design is full rank and has condition number no larger than normal.equations.max.condition,                       |

and otherwise falls back to SVD. "normal.equations" requests the normal-equation solve for full-rank local designs and falls back only on hard numerical failures. "svd" is the most stable reference path. "qr" uses pivoted QR and falls back to SVD for rank-deficient local designs.

normal.equations.max.condition

Positive condition-number guard used by local.solver = "auto" before accepting the normal-equation backend. The default is deliberately conservative because normal-equation solves square the effective condition number and order-3 near-minimum local supports can be numerically fragile.

ridge

Nonnegative diagonal ridge added to the linear system for numerical stabilization.

return.A

Logical. If TRUE, return *A* as a sparse matrix.

return.local.diagnostics

Logical. If TRUE, compute additional R-side local chart diagnostics such as chart distortion and boundary asymmetry. These diagnostics are useful for operator audits, but can be skipped in fitting and cross-validation paths for speed.

return.timing

Logical. If TRUE, attach a phase-level elapsed time table to the returned operator. The timings separate R-side validation, neighborhood/support construction, native local operator construction, optional local diagnostics, sparse matrix assembly, and output finalization. For adaptive-radius neighborhoods, subphase timings are also attached in neighborhoods\$timing.

verbose

Logical. If TRUE, print a short native construction message.

lambda.selection

Penalty selection: "fixed" (the default) uses lambda1 and lambda2; "cv" holds out observed labels; "gcv" uses generalized cross-validation. Grid inputs require an explicit "cv" or "gcv" selection mode.

lambda1.grid, lambda2.grid

Nonnegative penalty candidates for CV or GCV. lambda1.grid is required in either selection mode; lambda2.grid = NULL means 0. Do not also supply scalar lambda1 or lambda2. Unless explicitly set, stabilizer is enabled when any supplemental penalty candidate is positive.

cv.control

Named list used only with lambda.selection = "cv". Entries: foldid = NULL (one fold label per row of X; nonpositive or missing labels are ignored), cv.folds = 5L, loss = "mse" (or "mae"), and selection = "min" (or "one.se", choosing the largest total penalty within one standard error of the minimum). Support-search entries are support.selection = "rule" (or "cv"), support.grid = NULL, and support.max.candidates = 8L. Fold count is ignored when foldid is supplied.

gcv.control

Named list used only with lambda.selection = "gcv". Entries: trace.method = "exact" (or "hutchinson"), trace.n.probes = 50L, trace.seed = NULL, support.selection = "rule" (or "gcv"), support.grid = NULL, and support.max.candidates = 8L. Unknown, unnamed, or duplicated control entries are errors. Controls for a different selection mode are rejected rather than ignored.

## Details

For each response column, this function solves

$$(W + \lambda_1 B + \lambda_2 B_S + \epsilon I) \hat{f} = W y,$$

where  $W$  is the diagonal matrix of observation weights and  $\epsilon$  is ridge. Fully observed data with `lambda1 = lambda2 = ridge = 0` therefore reproduce the observed response exactly. Missing responses are excluded from the data-fit term by setting their weights to zero. The Matlab SSRHE semi-supervised convention is represented by a 0/1 labeled-indicator weights vector, or equivalently by setting unlabeled responses to NA: labeled vertices contribute unit diagonal data-fit terms and unlabeled vertices contribute no data-fit term.

The lower-level operator is matched to the Kim–Steinke–Hein SSRHE Matlab construction: self-including kNN neighborhoods, local PCA charts, a fixed-intercept local quadratic fit, doubled diagonal Hessian components, and optional supplemental stabilizer. The package exposes both  $A$  and  $B = A^\top A$ ; the fitted  $\ell_2$  estimator uses  $B$ .

## Value

A list of class `"ssrhe.hessian.fit"` containing fitted values, residuals, input response, weights, lambda parameters, objective/energy diagnostics, the reused operator, solver metadata, the call, and `lambda.selection`. CV adds class `"ssrhe.hessian.cv.fit"`, `cv.table`, `fold.id`, and `selection`; GCV adds class `"ssrhe.hessian.gcv.fit"`, `gcv.table`, `gcv.trace`, and `selection`. Support searches retain `selected.support` and the corresponding `support.cv.table` or `support.gcv.table`.

## Penalty selection

Fixed fitting supports response vectors or matrices. CV and GCV currently select penalties for one response vector at a time. Label CV removes each validation fold from the data-fit term and scores its held-out observed positive-weight labels. It supports missing responses. GCV requires a fully observed response and strictly positive observation weights, and minimizes

$$\frac{n^{-1} \sum_i w_i (y_i - \hat{f}_i)^2}{(1 - \text{tr}(S_\lambda)/n)^2},$$

where  $S_\lambda = (W + \lambda_1 B + \lambda_2 B_S + \epsilon I)^{-1} W$ . The exact trace is deterministic. The Hutchinson option uses Rademacher probes, reports trace standard errors, and is reproducible with `gcv.control = list(trace.method = "hutchinson", trace.seed = ...)`.

Both selection modes can also search adaptive-radius support profiles: set `support.selection` to the same mode in its control list. Each row of `support.grid` contains `adaptive.k.scale`, `min.support`, and optionally `max.support`. A NULL grid is constructed by `ssrhe.support.grid`. Each profile requires a new operator and a full penalty search, so this can substantially increase runtime.

## Migration

The former `fit.ssrhe.hessian.regression.cv()` and `fit.ssrhe.hessian.regression.gcv()` entry points have been retired. Use this function with `lambda.selection = "cv"` or `"gcv"`. Move `fold.id` to `cv.control$foldid`, `nfolds` to `cv.control$cv.folds`, and `loss/selection` to `cv.control`. Move `gcv.trace.method`, `gcv.trace.n.probes`, and `gcv.trace.seed` to `trace.method`, `trace.n.probes`,

and `trace.seed` in `gcv.control`. Support search settings also belong in the active control list; both former support candidate limits are now named `support.max.candidates`.

### Choosing controls

**First fit:** supply `X`, `y`, neighborhood size `k`, `tangent.dim`, and `lambda1`; use `lambda.selection = "fixed"` for a specified penalty.

**Selection:** `lambda.selection`, `lambda1.grid`, and `lambda2.grid` select penalties. Fold settings belong in `cv.control`; trace settings belong in `gcv.control`. Label CV and GCV are different criteria. Support selection rebuilds the operator for each candidate and multiplies this work.

**Geometry:** neighborhood, adaptive-radius, tangent and derivative controls determine the operator. Reuse it when fitting multiple responses.

**Numerics:** ridge and local conditioning controls stabilize solves. Exact GCV traces require more solves than stochastic trace estimates; the latter report sampling uncertainty and need a reproducible probe seed.

**Diagnostics:** inspect the selection table, energies, observed-label mask, and solver metadata. `fitted()` and `residuals()` preserve response matrices.

### References

Kim, K. I., Steinke, F., and Hein, M. (2009). Semi-supervised regression using Hessian energy with an application to semi-supervised dimensionality reduction. *Advances in Neural Information Processing Systems* 22. [https://papers.nips.cc/paper\\_files/paper/2009/hash/f4552671f8909587cf485ea990207f3b.html](https://papers.nips.cc/paper_files/paper/2009/hash/f4552671f8909587cf485ea990207f3b.html)

### Examples

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
fit.ssrhe.hessian.regression(
  X, sin(2 * pi * X[, 1]), k = 6L, tangent.dim = 1L, lambda1 = 0.1
)
fit.ssrhe.hessian.regression(
  X, sin(2 * pi * X[, 1]), k = 6L, tangent.dim = 1L,
  lambda.selection = "cv", lambda1.grid = c(0.01, 0.1),
  cv.control = list(cv.folds = 2L)
)
fit.ssrhe.hessian.regression(
  X, sin(2 * pi * X[, 1]), k = 6L, tangent.dim = 1L,
  lambda.selection = "gcv", lambda1.grid = c(0.01, 0.1)
)
```

## Description

Convenience wrapper that converts subject visit indices into a density input. Density-native methods dispatch to `fit.density`. LPS/PS-LPS methods fit ordinary smoother objects first and then call `normalize.density`; they are subject-occupation workflows, not standalone density-native methods.

## Usage

```
fit.subject.od(
  X,
  subject.index,
  method = c("empirical", "graph_random_walk", "lps_count", "ps_lps_count",
    "lps_logistic_binary", "chart_kernel", "local_likelihood_density",
    "local_likelihood_bernoulli"),
  graph = NULL,
  graph.control = list(),
  od.control = list(),
  return.details = TRUE,
  od.cv = c("none", "visit"),
  visit.foldid = NULL,
  visit.cv.folds = 5L,
  visit.cv.seed = 1L,
  visit.cv.epsilon = 1e-15,
  ...
)
```

## Arguments

|                             |                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>              | Numeric matrix with one row per support point.                                                                                                                                                                                                                                                                                                           |
| <code>subject.index</code>  | Integer row indices of subject visits in <code>X</code> . Repeated indices are allowed.                                                                                                                                                                                                                                                                  |
| <code>method</code>         | Density method identifier.                                                                                                                                                                                                                                                                                                                               |
| <code>graph</code>          | Optional precomputed graph object.                                                                                                                                                                                                                                                                                                                       |
| <code>graph.control</code>  | Named list for random walks: <code>walk.steps</code> (or <code>walk.step</code> ), <code>affinity.method</code> , <code>affinity.scale</code> , <code>affinity.epsilon</code> , and logical <code>normalize</code> . Underscore aliases are accepted, but conflicting aliases and unknown names are rejected. Must be empty for the empirical estimator. |
| <code>od.control</code>     | OD-facing alias for <code>density.control</code> .                                                                                                                                                                                                                                                                                                       |
| <code>return.details</code> | Logical; if <code>TRUE</code> , keep diagnostic details in the result.                                                                                                                                                                                                                                                                                   |
| <code>od.cv</code>          | OD-level selection mode. <code>"none"</code> preserves the direct fit. <code>"visit"</code> holds out subject visits, fits candidates on the remaining visits, and selects the candidate minimizing held-out negative log occupation mass.                                                                                                               |
| <code>visit.foldid</code>   | Optional positive integer vector assigning subject visits to OD-level cross-validation folds. Its length must equal <code>length(subject.index)</code> .                                                                                                                                                                                                 |
| <code>visit.cv.folds</code> | Number of visit folds when <code>visit.foldid</code> is not supplied.                                                                                                                                                                                                                                                                                    |
| <code>visit.cv.seed</code>  | Random seed for generated visit folds.                                                                                                                                                                                                                                                                                                                   |

```
visit.cv.epsilon
      Positive floor used in held-out  $-\log(\rho[\text{visit}])$  scoring.
...
      Reserved for future methods; unused arguments are errors.
```

### Details

The "lps\_logistic\_binary" method name is historical. In the current OD workflow it calls `fit.lps(..., outcome.family = "bernoulli")`, which fits a clipped probability field with the LPS least-squares core and then converts that field to a density with `normalize.density`. It is not the `outcome.family = "binomial"` local-logistic IRLS path.

When `od.cv = "visit"`, graph random-walk candidates may pass OD-level grids inside `graph.control`: `walk.step.grid`, `affinity.method.grid`, `affinity.scale.grid`, `affinity.epsilon.grid`, and `normalize.grid`. Missing `affinity.scale` values mean the usual data-derived affinity scale.

When `od.cv = "visit"`, chart-based and LPS-family methods may pass `chart.dim.grid` through .... For "chart\_kernel", "local\_likelihood\_density", "local\_likelihood\_bernoulli", "lps\_count", "lps\_logistic\_binary", and "ps\_lps\_count", this grid compares fixed integer chart dimensions with optional "auto" and "local.auto" chart-dimension policies under the same held-out-visit negative-log-occupation score. For LPS-family OD visit CV, each outer candidate is passed to the source smoother as a scalar local model configuration; this avoids nested row-level multi-candidate selection inside each held-out-visit fold.

### Value

A list of class "density\_fit". Its `rho` component is the estimated probability mass over the rows of `X`, and `empirical.rho` is the normalized visit-count mass. The `subject` component summarizes the number of visits, the number of distinct visited support points, the largest visit multiplicity, and the fraction of repeatedly visited support points. Other components describe the fitted method, normalization accounting, smoothing diagnostics, and warnings as documented in `fit.density`. With `od.cv = "visit"` and `return.details = TRUE`, the result also contains the visit-level cross-validation table, fold assignments, and held-out predicted masses.

### Examples

```
X <- matrix(seq(0, 1, length.out = 6), ncol = 1)
fit.subject.od(X, subject.index = c(1L, 3L, 3L, 6L),
  method = "empirical")
```

---

|                     |                                                     |
|---------------------|-----------------------------------------------------|
| get.region.boundary | <i>Get Boundary Vertices of a Region in a Graph</i> |
|---------------------|-----------------------------------------------------|

---

### Description

Identifies the boundary vertices of a specified region in a graph, defined as vertices in the region that have at least one neighbor outside the region.

### Usage

```
get.region.boundary(adj.list, region)
```

**Arguments**

|                       |                                                                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list</code> | A list of integer vectors, where each vector contains the indices of vertices adjacent to the corresponding vertex. Indices must be 1-based. |
| <code>region</code>   | An integer vector of vertex indices (1-based) defining the region for which to find boundary vertices.                                       |

**Details**

This function determines which vertices in a specified region are positioned at the boundary - meaning they have at least one neighbor that is not part of the region. These boundary vertices are often treated differently in graph-based algorithms, such as harmonic smoothing, where their values are typically fixed as constraints.

The boundary definition used matches the one implemented in the C++ harmonic smoothing functions, focusing on vertices that have connections to the outside of the region.

**Value**

An integer vector containing the indices of the boundary vertices, which is a subset of the input region vector. Returns an empty integer vector if no boundary vertices exist.

**See Also**

[perform.harmonic.smoothing](#), [harmonic.smoother](#)

**Examples**

```
# Create a simple grid graph adjacency list (5x5 grid)
create_grid_adj_list <- function(n_rows, n_cols) {
  n <- n_rows * n_cols
  adj_list <- vector("list", n)
  for (i in 1:n_rows) {
    for (j in 1:n_cols) {
      v <- (i-1) * n_cols + j
      neighbors <- numeric(0)

      # Add neighbors (up, down, left, right)
      if (i > 1) neighbors <- c(neighbors, (i-2) * n_cols + j) # up
      if (i < n_rows) neighbors <- c(neighbors, i * n_cols + j) # down
      if (j > 1) neighbors <- c(neighbors, (i-1) * n_cols + (j-1)) # left
      if (j < n_cols) neighbors <- c(neighbors, (i-1) * n_cols + (j+1)) # right

      adj_list[[v]] <- neighbors
    }
  }
  return(adj_list)
}

# Create a 5x5 grid graph
grid_adj_list <- create_grid_adj_list(5, 5)
```

```
# Define a region (central 3x3 subgrid)
central_region <- c(7:9, 12:14, 17:19)

# Find boundary vertices of the central region
boundary <- get.region.boundary(grid_adj_list, central_region)
print(boundary)
# Expected output: c(7, 8, 9, 12, 14, 17, 18, 19)
# (all except the center vertex 13)
```

---

graph.trend.filtering.operator

*Construct a Graph Trend-Filtering Operator*

---

## Description

Builds the weighted graph difference operators used by graph trend filtering. The default operator.family = "graph.laplacian.recursive" uses recursive graph operators

$$k = 0 : \quad \Delta_w^{(1)} = D_w,$$

$$k = 1 : \quad \Delta_w^{(2)} = L_w = D_w^\top D_w,$$

and

$$k = 2 : \quad \Delta_w^{(3)} = D_w L_w.$$

The experimental operator.family = "path.divided.difference" builds canonical path rows and computes position-aware finite differences along graph paths using weight.list as metric edge lengths.

## Usage

```
graph.trend.filtering.operator(
  adj.list,
  weight.list = NULL,
  order = 0L,
  weight.rule = c("conductance", "sqrt.conductance", "unit"),
  operator.family = c("graph.laplacian.recursive", "path.divided.difference"),
  path.family = c("branch.continuation", "all.simple"),
  path.weighting = c("unit", "anchor.mean"),
  return.sparse = TRUE
)
```

## Arguments

**adj.list** List of integer neighbor vectors using 1-based vertex indices. The graph must be undirected: every edge must appear in both endpoint adjacency lists.

|                 |                                                                                                                                                                                                                                                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| weight.list     | Optional list of positive edge weights parallel to adj.list. For operator.family = "graph.laplacian.recursive", weights are graph trend-filtering weights or conductances. For operator.family = "path.divided.difference", weights are metric edge lengths used to define path coordinates. If NULL, all values are one. |
| order           | Integer trend-filtering order. Supported values are 0L, 1L, and 2L.                                                                                                                                                                                                                                                       |
| weight.rule     | Character scalar. "conductance" uses weight.list directly as $\omega_e$ ; "sqrt.conductance" uses $\sqrt{w_e}$ ; "unit" ignores weight.list. This applies to operator.family = "graph.laplacian.recursive".                                                                                                               |
| operator.family | Character scalar selecting the penalty operator family. The default "graph.laplacian.recursive" preserves the existing graph trend-filtering semantics. "path.divided.difference" uses local pathwise finite differences and is designed to reproduce classical one-dimensional trend filtering on path graphs.           |
| path.family     | Character scalar used for operator.family = "path.divided.difference". "branch.continuation" keeps paths whose internal vertices do not pass through branch points. "all.simple" keeps all simple paths of the required length.                                                                                           |
| path.weighting  | Character scalar used for operator.family = "path.divided.difference". "unit" assigns every canonical path row weight one. "anchor.mean" divides rows by the number of retained canonical paths sharing the same canonical start vertex.                                                                                  |
| return.sparse   | Logical. If TRUE, attach <b>Matrix</b> sparse incidence, Laplacian, and penalty matrices.                                                                                                                                                                                                                                 |

### Value

A list of class "graph.trend.filtering.operator" containing the canonical graph, edge table, trend-filtering weights, incidence, weighted Laplacian, selected penalty operator, and nullity estimate.

### Examples

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
graph.trend.filtering.operator(adj, order = 1L, weight.rule = "unit")
```

---

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| harmonic.smoother | <i>Perform Harmonic Smoothing with Topology Tracking</i> |
|-------------------|----------------------------------------------------------|

---

### Description

Relaxes graph values while recording changes in local extrema. Boundary vertices have neighbors outside the region and retain their original values.

**Usage**

```

harmonic.smoother(
  adj.list,
  weight.list,
  values,
  region.vertices,
  max.iterations = 100,
  tolerance = 1e-06,
  record.frequency = 1,
  stability.window = 3,
  stability.threshold = 0.05
)

```

**Arguments**

|                                  |                                                                                                                                          |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list</code>            | A list of integer vectors, where each vector contains indices of vertices adjacent to the corresponding vertex. Indices must be 1-based. |
| <code>weight.list</code>         | A list of numeric vectors containing weights of edges corresponding to adjacencies in <code>adj.list</code> .                            |
| <code>values</code>              | A numeric vector of function values defined at each vertex.                                                                              |
| <code>region.vertices</code>     | An integer vector of vertex indices (1-based) defining the region to be smoothed. Boundary vertices will have fixed values.              |
| <code>max.iterations</code>      | Integer scalar, the maximum number of relaxation iterations to perform. Default is 100.                                                  |
| <code>tolerance</code>           | Numeric scalar, the convergence threshold for value changes. Default is 1e-6.                                                            |
| <code>record.frequency</code>    | Integer scalar, how often to record states (every N iterations). Default is 1 (record every iteration).                                  |
| <code>stability.window</code>    | Integer scalar, number of consecutive recorded differences to check for extrema stability. Default is 3.                                 |
| <code>stability.threshold</code> | Numeric scalar in $[0, 1]$ , maximum allowed difference in topology to consider stable. Default is 0.05.                                 |

**Details**

Uses the same inverse-length neighbor averages and boundary definition as [perform.harmonic.smoothing](#). Unlike the basic function, it also relaxes regions with no boundary. Such a problem is not uniquely anchored; synchronous updates can oscillate on bipartite graphs, including paths.

The recorded diagnostic compares sets of (vertex, minimum/maximum) pairs. Its value is one minus twice the number of shared pairs divided by the sum of the two set sizes (zero for two empty sets). A detection means that the last `stability.window` recorded differences are at most `stability.threshold`. It does not establish numerical convergence, optimal smoothing, noise removal, or stability of full attraction basins. In particular, recording every two iterations can conceal an alternating oscillation. Always inspect converged and `max_residual` separately. Stability

detection does not stop relaxation or select the returned fit. The initial and final states are always recorded, including a final partial recording interval. Disconnected components without fixed boundary values have the same lack of anchoring as a region without a boundary.

### Value

A list of class "harmonic\_smoother" containing:

|                                        |                                                                                                  |
|----------------------------------------|--------------------------------------------------------------------------------------------------|
| harmonic_predictions                   | Numeric vector of smoothed function values                                                       |
| i_harmonic_predictions                 | Matrix of function values at each recorded iteration (columns are iterations)                    |
| i_basins                               | List of matrices representing extrema at each iteration                                          |
| stable_iteration                       | First iteration with a detected extrema-stability window, or NA_integer_ if none was detected    |
| stability_detected                     | Whether an extrema-stability window was detected                                                 |
| num_iterations, recorded_iterations    | Number of relaxation updates and their actual indices for the recorded columns, starting at zero |
| max_change, max_residual               | Per-update maximum change and harmonic residual. Both must fall below tolerance for convergence  |
| status                                 | converged, iteration_limit, or no_interior                                                       |
| topology_differences                   | Numeric vector of differences between consecutive recorded iterations                            |
| basin_cx_differences                   | Alias of topology_differences                                                                    |
| converged                              | Logical indicator of whether the relaxation converged                                            |
| num_region, num_boundary, num_interior | Vertex counts for the requested region and its boundary/interior split                           |

### See Also

[perform.harmonic.smoothing](#) for basic smoothing, [plot.harmonic\\_smoother](#) for visualization methods, [summary.harmonic\\_smoother](#) for summary statistics

### Examples

```
adj.list <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
weight.list <- list(1, c(1, 1), c(1, 1), 1)
values <- c(0, 2, -1, 1)
result <- harmonic.smoother(
  adj.list, weight.list, values, region.vertices = seq_along(values),
  max.iterations = 20, record.frequency = 2
)
smoothed.values <- result$harmonic_predictions
smoothed.values
```

lpl.tf.operator

*Construct a Local Polynomial Lifting Trend-Filtering Operator***Description**

Builds the Local Polynomial Lifting Trend Filtering (LPL-TF) analysis operator from local polynomial prediction residuals. For each observed target point  $i$ , the operator predicts  $f_i$  from nearby values  $f_j$  using a local polynomial model and stores the residual row

$$r_i(f) = f_i - \sum_{j \in S_i} h_{ij} f_j.$$

**Usage**

```
lpl.tf.operator(
  X,
  adj.list = NULL,
  weight.list = NULL,
  graph = NULL,
  graph.stage = "final",
  anchor.index = NULL,
  anchor.coordinates = NULL,
  degree = 2L,
  support.type = c("adaptive.radius", "knn", "fixed.radius"),
  support.size = NULL,
  radius = NULL,
  min.support = NULL,
  support.buffer = 3L,
  kernel = c("epanechnikov", "triangular", "gaussian", "tricube"),
  coordinate.method = c("coordinates", "local.pca"),
  chart.dim = NULL,
  support.metric = c("auto", "coordinates", "graph.geodesic"),
  auto.chart.support.metric = c("coordinates", "operator", "both"),
  auto.chart.selection.metric = c("coordinates", "operator"),
  exclude.self = TRUE,
  row.normalize = c("l2", "none", "l1"),
  local.solver = c("auto", "normal.equations", "qr", "svd"),
  normal.equations.max.condition = 1e+08,
  duplicate.action = c("keep", "error"),
  drop.rank.deficient = TRUE,
  verbose = FALSE,
  ...
)
```

**Arguments**

**X** Numeric coordinate matrix with one row per observation.

|                                          |                                                                                                                                                                                                                                                                                                                             |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list, weight.list</code>       | Optional supplied undirected graph adjacency and positive edge-length lists using 1-based vertex indices.                                                                                                                                                                                                                   |
| <code>graph</code>                       | Optional graph object returned by <code>dgraphs::create.rknn.graph()</code> or a list containing <code>adj.list/weight.list</code> or <code>adj_list/weight_list</code> .                                                                                                                                                   |
| <code>graph.stage</code>                 | Graph stage to extract from graph: "final", "raw", or "pruned".                                                                                                                                                                                                                                                             |
| <code>anchor.index</code>                | Optional integer vector of observed anchor/target indices. The current implementation supports observed anchors only.                                                                                                                                                                                                       |
| <code>anchor.coordinates</code>          | Reserved for later off-observation anchors; this implementation requires NULL.                                                                                                                                                                                                                                              |
| <code>degree</code>                      | Polynomial degree. The current implementation supports 0L, 1L, and 2L.                                                                                                                                                                                                                                                      |
| <code>support.type</code>                | Coordinate support rule. "knn" uses the nearest <code>support.size</code> candidates including the target before self exclusion. "adaptive.radius" uses the nearest <code>min.support</code> positive predictors after self exclusion. "fixed.radius" uses all points within radius.                                        |
| <code>support.size</code>                | Positive integer for <code>support.type = "knn"</code> .                                                                                                                                                                                                                                                                    |
| <code>radius</code>                      | Positive coordinate radius for <code>support.type = "fixed.radius"</code> .                                                                                                                                                                                                                                                 |
| <code>min.support</code>                 | Minimum number of positive-weight predictors after self exclusion. If NULL, uses <code>choose(ncol(X) + degree, degree) + support.buffer</code> .                                                                                                                                                                           |
| <code>support.buffer</code>              | Nonnegative integer added to the local polynomial design size when choosing default supports.                                                                                                                                                                                                                               |
| <code>kernel</code>                      | Kernel used to weight local predictors by distance.                                                                                                                                                                                                                                                                         |
| <code>coordinate.method</code>           | Coordinate chart used for the local polynomial design. "coordinates" uses centered ambient coordinates; "local.pca" uses a deterministic local PCA chart on the selected support.                                                                                                                                           |
| <code>chart.dim</code>                   | Chart dimension. For "coordinates", this must be NULL or <code>ncol(X)</code> . For "local.pca", NULL defaults to <code>ncol(X)</code> . The special value "auto" estimates a single chart dimension from observed-coordinate local PCA spectra only, without using responses, truth values, latent coordinates, or labels. |
| <code>support.metric</code>              | Support-distance rule. "coordinates" uses Euclidean coordinate distances; "graph.geodesic" uses shortest-path distances in the supplied graph. "auto" uses graph geodesics when a graph is supplied and coordinate distances otherwise.                                                                                     |
| <code>auto.chart.support.metric</code>   | Support system used when <code>chart.dim = "auto"</code> . "coordinates" uses Euclidean coordinate neighborhoods, "operator" uses the resolved operator support metric, and "both" computes both diagnostics side by side.                                                                                                  |
| <code>auto.chart.selection.metric</code> | Which auto chart-dimension diagnostic to use for the fitted operator when <code>auto.chart.support.metric = "both"</code> . The default "coordinates" preserves historical behavior.                                                                                                                                        |
| <code>exclude.self</code>                | Logical. LPL-TF residual rows require TRUE.                                                                                                                                                                                                                                                                                 |
| <code>row.normalize</code>               | Complete-row normalization rule applied after setting target coefficient +1 and predictor coefficients <code>-h<sub>ij</sub></code> .                                                                                                                                                                                       |

|                                |                                                                             |
|--------------------------------|-----------------------------------------------------------------------------|
| local.solver                   | Local linear algebra rule.                                                  |
| normal.equations.max.condition | Maximum condition number for local.solver = "auto" to use normal equations. |
| duplicate.action               | Duplicate coordinate policy.                                                |
| drop.rank.deficient            | Logical. The current implementation requires TRUE.                          |
| verbose                        | Logical. Reserved for diagnostic messages.                                  |
| ...                            | Reserved.                                                                   |

### Details

The operator uses observed anchors, excludes the target point from every predictor support, and drops requested-degree rank-deficient rows with explicit metadata. There is no silent degree downgrade. Graph support distances affect support selection and kernel bandwidths only; local polynomial coordinates are controlled separately by `coordinate.method`. Graph construction is supplied by **dgraphs** or by explicit `adj.list/weight.list` payloads. Coordinate support paths and graph-geodesic payload validation are package-local; shortest-path distances are computed through **dgraphs**.

If  $a_i$  is the complete assembled residual row, then `row.normalize = "l2"` uses  $a_i/\|a_i\|_2$ , and `row.normalize = "l1"` uses  $a_i/\|a_i\|_1$ . Normalization is applied to the whole row, not only to predictor coefficients.

### Value

A list of class "lpl\_tf\_operator" containing the sparse operator matrix A, row metadata, supports, local design summaries, diagnostics, settings, and the matched call.

### Examples

```
X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
lpl.tf.operator(X, degree = 1L, support.type = "knn",
               support.size = 7L, kernel = "gaussian")
```

---

lps.backend.diagnostics

*Report LPS Backend and Chart-Dimension Diagnostics*

---

### Description

Builds a compact one-row diagnostic table for a fitted local polynomial smoother. The table records the requested backend, the backend actually used, the requested chart-dimension rule, the resolved chart dimension, selected tuning parameters, and whether the fit follows the current deployable local-PCA auto-dimension contract.

**Usage**

```
lps.backend.diagnostics(object)
```

**Arguments**

object                      A fitted "lps" object.

**Details**

The current backend policy is conservative: `backend = "auto"` uses the C++ backend for ambient-coordinate LPS, but uses the R reference backend for `coordinate.method = "local.pca"`. The native local-PCA backend `"cpp.local.pca"` remains an explicit opt-in backend. This helper makes that policy visible in reports and downstream experiment manifests without changing the default.

For real-data local-PCA runs, the deployable chart-dimension contract is the ordinary `local.chart.method = "pca"` path with `chart.dim = "auto"` or `"local.auto"` and observed-covariate auto-dimension diagnostics. In P7-style experiments this is paired with `auto.chart.support.metric = "both"` and `auto.chart.selection.metric = "operator"`. The experimental `"second.order.svd"` chart path is reported explicitly but is not certified by this deployable local-PCA contract. For LPS itself, which uses coordinate supports, the operator-support diagnostic is currently equivalent to the coordinate-support diagnostic; the fields are still recorded so the same manifest schema can be shared with LPL-TF and S-LPL-TF experiments.

**Value**

A one-row `data.frame` with `backend`, `chart-dimension`, `selection`, `candidate-count`, and `policy` fields.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 16), ncol = 1)
fit <- fit.lps(X, X[, 1]^2, support.grid = 6L, degree.grid = 1L,
              kernel.grid = "gaussian", cv.folds = 2L, backend = "R")
lps.backend.diagnostics(fit)
```

---

|                                 |                                                   |
|---------------------------------|---------------------------------------------------|
| <code>lps.grouped.foldid</code> | <i>Build a Cluster-Respecting Fold Assignment</i> |
|---------------------------------|---------------------------------------------------|

---

**Description**

Assigns whole clusters to cross-validation folds so that no cluster is split across folds: every observation of a cluster receives the same fold id, and therefore no training set ever shares a cluster with the fold held out against it. Folds are built by a deterministic size-balanced greedy rule: clusters are taken largest first (ties by first appearance in `cluster.id`) and each is placed into the currently smallest fold (ties by lowest fold index). With `v` equal to the number of clusters this reduces to leave-cluster-out.

**Usage**

```
lps.grouped.foldid(cluster.id, v = 5L, shuffle.seed = NULL)
```

**Arguments**

|                           |                                                                                                        |
|---------------------------|--------------------------------------------------------------------------------------------------------|
| <code>cluster.id</code>   | Vector of cluster labels (factor, character, or integer-like), one per observation, no missing values. |
| <code>v</code>            | Number of folds; an integer between 2 and the number of distinct clusters.                             |
| <code>shuffle.seed</code> | Optional integer seed for a randomized cluster order; NULL (default) keeps the deterministic order.    |

**Details**

The assignment is fully deterministic when `shuffle.seed` is NULL. Supplying `shuffle.seed` applies one seeded permutation to the cluster order before the greedy pass (the seed is consumed immediately via `set.seed`), giving a reproducible randomized variant; callers following the LPS evidence conventions should record the seed they pass.

**Value**

An integer fold-id vector of length `length(cluster.id)` with values in `1:v`; every fold is nonempty and every cluster maps to exactly one fold.

**See Also**

[lps.nested.cv\(\)](#) for nested cross-validation that can consume grouped folds at both the outer and inner level.

**Examples**

```
cluster <- rep(letters[1:6], each = 2)
lps.grouped.foldid(cluster, v = 3L)
```

---

lps.nested.cv

---

*Nested Cross-Validation for LPS with Explicit, Recorded Folds*


---

**Description**

Runs outer-fold nested cross-validation around [fit.lps\(\)](#): for each outer fold, candidate selection is one ordinary `fit.lps` call on the inner-training rows only, with an explicit inner fold id and `X.eval` set to the held-out outer-test rows, so the held-out fold never participates in inner selection. The pooled outer-test error is the nested generalization estimate. The same call also computes the *selected-min* arm — an ordinary `fit.lps` on all rows using the *same* `outer.foldid` — so optimism comparisons are paired on one fold assignment by construction.

**Usage**

```
lps.nested.cv(
  X,
  y,
  outer.foldid,
  fit.args = list(),
  inner.folds = 5L,
  cluster.id = NULL,
  inner.foldid.method = c("round.robin", "grouped"),
  inner.shuffle.seed = NULL
)
```

**Arguments**

|                                  |                                                                                                                                                                                                                                                                                                      |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                   | Numeric matrix of observations (rows) used for training.                                                                                                                                                                                                                                             |
| <code>y</code>                   | Numeric response vector with <code>length(y) == nrow(X)</code> .                                                                                                                                                                                                                                     |
| <code>outer.foldid</code>        | Positive integer vector of length <code>nrow(X)</code> assigning rows to outer folds (at least two distinct folds).                                                                                                                                                                                  |
| <code>fit.args</code>            | Named list of additional arguments passed to every <code>fit.lps</code> call (grids, back-end, design settings, ...). Must not contain <code>X</code> , <code>y</code> , <code>foldid</code> , or <code>X.eval</code> : the fold plumbing is owned by this function so both arms see the same folds. |
| <code>inner.folds</code>         | Number of inner selection folds (integer $\geq 2$ ).                                                                                                                                                                                                                                                 |
| <code>cluster.id</code>          | Optional cluster labels (length <code>nrow(X)</code> ), required for <code>inner.foldid.method = "grouped"</code> .                                                                                                                                                                                  |
| <code>inner.foldid.method</code> | Inner fold construction: <code>"round.robin"</code> (default) or <code>"grouped"</code> (cluster-respecting).                                                                                                                                                                                        |
| <code>inner.shuffle.seed</code>  | Optional integer; per-fold offset seed for the randomized variants described above. Recorded in the return value.                                                                                                                                                                                    |

**Details**

The existing `fit.lps` behavior is untouched: this utility consumes the public API with explicit `foldid` only. It currently supports `outcome.family = "gaussian"` (the default) and scores with RMSE.

Inner folds are built per outer fold and fully recorded in the return value: `"round.robin"` assigns `rep_len(1:inner.folds, n)` over the inner-training rows in row order (deterministic; if `inner.shuffle.seed` is supplied, fold `k` — by position — uses `set.seed(inner.shuffle.seed + k)` and permutes the balanced assignment, giving a reproducible randomized variant); `"grouped"` builds cluster-respecting inner folds with `lps.grouped.foldid()` on the inner-training rows of `cluster.id` (passing the same per-fold offset seed when `inner.shuffle.seed` is supplied).

**Value**

A list of class `"lps_nested_cv"`:

`nested.rmse` pooled RMSE of the outer-test predictions over all rows with finite predictions.

`n.missing.predictions` count of non-finite outer-test predictions (e.g. `unstable.action = "na"` fallbacks).

`folds` one row per outer fold: fold label, test size, selected `support.size / degree / kernel / bandwidth.multiplier`, the inner selected-min CV score, and the fold's outer-test RMSE.

`predictions` `length-nrow(X)` vector of outer-test predictions (each row predicted by the fold that held it out).

`selected.min` the paired selected-min arm on the same outer `foldid`: selected (the `fit.lps` selected row), `cv.score` (its observed CV RMSE), `foldid`, and `fit` (the full-data `fit.lps` object, usable for deployment on an external test set).

`outer.foldid`, `test.index`, `train.index`, `inner.foldid`, `inner.foldid.used`, `inner.cv.table` complete fold/index telemetry: the realized inner fold ids as constructed and as recorded by each inner `fit.lps` (`$foldid`), per-fold index sets, and each fold's full inner CV candidate table.

`outer.cluster.whole` TRUE/FALSE whether every cluster lies wholly inside one outer fold (NA when `cluster.id` is absent).

`inner.folds`, `inner.foldid.method`, `inner.shuffle.seed`, `fit.args`, `call` the recorded configuration.

### See Also

[lps.grouped.foldid\(\)](#) for the grouped fold constructor.

### Examples

```
X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
y <- sin(2 * pi * X[, 1])
nested <- lps.nested.cv(
  X, y, outer.foldid = rep(1:3, length.out = 18), inner.folds = 2L,
  fit.args = list(support.grid = 6L, degree.grid = 1L,
    kernel.grid = "gaussian", backend = "R")
)
nested$rmse
```

---

|                                 |                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------|
| <code>lps.pointwise.band</code> | <i>Pointwise Variance and Confidence Band for a Fixed-Configuration LPS Fit</i> |
|---------------------------------|---------------------------------------------------------------------------------|

---

### Description

Computes, from the analytically extracted linear-smoother matrix  $S$  of a fixed-configuration LPS fit (see [smoother.matrix\(\)](#)), the pointwise variance  $\text{Var}(\text{fitted}_i) = \sigma^2 * \sum_j S_{ij}^2$ , the effective degrees of freedom  $\text{df} = \text{tr}(S)$ , the plug-in noise estimate  $\hat{\sigma}^2 = \text{RSS} / (n - 2 * \text{tr}(S) + \sum(S^2))$ , and the pointwise confidence band  $\text{fitted}_i \pm z * \sigma * ||S_{i.}||_2$  with  $z = \text{qnorm}(1 - (1 - \text{level}) / 2)$ .

**Usage**

```
lps.pointwise.band(
  object,
  sigma = NULL,
  level = 0.95,
  check.tol = 1e-10,
  variance.method = c("residual", "legacy")
)
```

**Arguments**

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object          | A fitted "lps" object from <code>fit.lps()</code> at a fixed configuration, with <code>X.eval</code> identical to <code>X</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| sigma           | Known noise standard deviation (positive scalar), or <code>NULL</code> (default) to use the plug-in <code>sigma.hat</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| level           | Confidence level of the band, a single number strictly between 0 and 1. Default 0.95. For independent errors with common variance and a fixed smoother, the expected residual sum of squares is $\sigma^2 * \sum((I - S)^2) +   (I - S) \mu  ^2$ . The default denominator is therefore the residual noise degrees of freedom, $n - 2 * \text{tr}(S) + \sum(S^2)$ , computed as $\sum((I - S)^2)$ for stability. This corrects the variance contribution; it cannot remove signal bias. Normal bands target the smoothed mean $S \mu$ . Their interpretation as bands for $\mu$ requires negligible smoothing bias. Plug-in bands are approximate, pointwise, and conditional on the specified configuration; they do not account for tuning on the same responses, dependent errors, or simultaneous coverage. |
| check.tol       | Passed to <code>smoother.matrix()</code> 's self-guard.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| variance.method | Noise-variance denominator: "residual" (default) uses $\sum((I - S)^2)$ ; "legacy" reproduces the former $n - \text{tr}(S)$ rule. Neither removes smoothing bias. Added after <code>check.tol</code> to preserve positional calls. The choice affects <code>sigma.hat</code> even when a known <code>sigma</code> is supplied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**Details**

With `sigma` supplied (known noise standard deviation), the band uses it directly (`sigma.source = "known"`); with `sigma = NULL` the plug-in `sigma.hat` is used (`sigma.source = "plug.in"`). `sigma.hat` is reported in both modes. The variance is purely the linear-smoother sampling variance: the band makes no bias correction, so where bias dominates (boundary, high curvature) undercoverage is expected and must be reported, not masked.

Requires `X.eval` identical to `X` (the square training-fit smoother, on which `tr(S)` and `RSS` are defined), in addition to all restrictions of `smoother.matrix()`. If any evaluation point's local fit returned `NA`, its variance/band entries are `NA`, and `df`, `rss`, and `sigma.hat` are `NA` as well; supplying a known `sigma` still yields bands at the unaffected points.

**Value**

A list of class "lps.pointwise.band" with named fields: `fitted` (the fit's raw fitted values), `se`, `variance`, `lower`, `upper`, `level`, `z`, `sigma` (the supplied known `sigma`, or `NA` in plug-in mode),

sigma.hat, sigma.source ("known" or "plug.in"), df (tr(S)), residual.df, variance.method, rss, n.train, smoother.row.norm (||S\_i.||\_2), and configuration (the pinned fit configuration).

### See Also

[smoother.matrix\(\)](#)

### Examples

```
set.seed(1)
n <- 30
X <- matrix(runif(2 * n, -1, 1), ncol = 2)
y <- sin(pi * X[, 1]) + 0.1 * rnorm(n)
fit <- fit.lps(X, y, foldid = rep(1:2, length.out = n),
              support.grid = 12L, degree.grid = 1L,
              kernel.grid = "tricube", backend = "R",
              design.basis = "orthogonal.polynomial.drop",
              ridge.multiplier.grid = 0, ridge.condition.max = Inf,
              unstable.action = "na")
band.known <- lps.pointwise.band(fit, sigma = 0.1)
band.plugin <- lps.pointwise.band(fit)
band.known$df
band.plugin$sigma.hat
```

---

malps.gcv

---

*Compute MALPS Linear-Smoother Diagnostics*


---

### Description

Computes effective degrees of freedom, generalized cross-validation (GCV), and analytic leave-one-out residual diagnostics from a MALPS smoother matrix. These diagnostics are exact for fixed-support, fixed-weight linear MALPS fits. For cross-validated fits they are conditional on the selected support profile. Robust fits are rejected by default for the same reason described in [smoother.matrix](#).

### Usage

```
malps.gcv(
  object,
  y = NULL,
  smoother.matrix = NULL,
  include.loocv = TRUE,
  max.n = 1000L,
  allow.robust = FALSE,
  ...
)
```

**Arguments**

|                              |                                                                                                              |
|------------------------------|--------------------------------------------------------------------------------------------------------------|
| <code>object</code>          | A "malps" object from <code>fit.malps</code> or <code>refit</code> .                                         |
| <code>y</code>               | Optional response vector. Defaults to <code>object\$y</code> .                                               |
| <code>smoother.matrix</code> | Optional precomputed matrix from <code>smoother.matrix</code> .                                              |
| <code>include.loocv</code>   | Logical; include analytic leave-one-out residuals and mean squared error.                                    |
| <code>max.n</code>           | Maximum dense smoother size passed to <code>smoother.matrix</code> when <code>smoother.matrix</code> = NULL. |
| <code>allow.robust</code>    | Logical; passed to <code>smoother.matrix</code> .                                                            |
| <code>...</code>             | Reserved for future extensions.                                                                              |

**Details**

The GCV score is computed as

$$\text{GCV} = \frac{n^{-1} \|y - Sy\|_2^2}{(1 - \text{tr}(S)/n)^2}.$$

The analytic leave-one-out residuals are computed as

$$e_i^{\text{loo}} = \frac{y_i - \hat{y}_i}{1 - S_{ii}}.$$

**Value**

A list with residual, fitted-value, EDF, GCV, and optional LOOCV diagnostics.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.malps(X, X[, 1]^2, degree = 1L,
               support.type = "knn", support.size = 8L)
malps.gcv(fit)$gcv
```

---

`materialize.synthetic` *Materialize a synthetic-data specification*

---

**Description**

Materialize a synthetic-data specification

**Usage**

```
materialize.synthetic(
  spec,
  n = NULL,
  seed,
  rng.policy = c("named.stream.v1", "legacy"),
  validate = TRUE
)
```

**Arguments**

|            |                                               |
|------------|-----------------------------------------------|
| spec       | A synthetic_spec.                             |
| n          | Sample size, or NULL for fixed-size sampling. |
| seed       | Scalar whole-number seed.                     |
| rng.policy | Versioned RNG policy.                         |
| validate   | Whether to validate the returned dataset.     |

**Value**

A synthetic\_dataset.

**Examples**

```
spec <- synthetic.registry.spec("G1")
materialize.synthetic(spec, n = 20L, seed = 1L)
```

---

materialize.synthetic.instance

*Materialize a frozen synthetic instance*

---

**Description**

Materialize a frozen synthetic instance

**Usage**

```
materialize.synthetic.instance(instance.id, validate = TRUE)
```

**Arguments**

|             |                                              |
|-------------|----------------------------------------------|
| instance.id | Frozen instance ID.                          |
| validate    | Whether to validate and checksum the result. |

**Value**

A synthetic\_dataset whose dataset.id is instance.id.

**Examples**

```
materialize.synthetic.instance("geosmooth.g1.default.v1")
```

---

metric.graph.heat.eta.grid

*Construct a Graph Heat-Time Grid*


---

## Description

Creates a positive heat-time grid from a "metric.graph.lowpass.basis" object.

## Usage

```
metric.graph.heat.eta.grid(
  basis,
  rule = c("spectral_guarded", "w1_inverse_spectrum"),
  n.initial = 40L,
  include.zero = FALSE,
  truncation.tol = 1e-04,
  equilibrium.tol = 1e-04
)
```

## Arguments

|                 |                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| basis           | A "metric.graph.lowpass.basis" object.                                                                                                                                                                                                                                                                                                                                       |
| rule            | Grid rule. "w1_inverse_spectrum" requires a complete basis and reproduces the W1 inverse-spectrum grid. "spectral_guarded" raises the lower endpoint for a truncated basis until the conservative omitted-mode attenuation bound is no larger than truncation.tol, and extends the upper endpoint until the slowest retained positive mode is attenuated to equilibrium.tol. |
| n.initial       | Number of positive grid values.                                                                                                                                                                                                                                                                                                                                              |
| include.zero    | Logical. Include the exact no-smoothing endpoint.                                                                                                                                                                                                                                                                                                                            |
| truncation.tol  | Positive tolerance smaller than one for the conservative omitted-mode attenuation bound.                                                                                                                                                                                                                                                                                     |
| equilibrium.tol | Positive tolerance smaller than one for the slowest positive retained mode at the upper endpoint.                                                                                                                                                                                                                                                                            |

## Value

A numeric vector with grid-construction metadata stored as attributes.

## Examples

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
lengths <- list(1, c(1, 1), c(1, 1), 1)
basis <- metric.graph.lowpass.basis(adj, lengths, n.eigenpairs = 4L,
                                   eigen.solver = "dense")
metric.graph.heat.eta.grid(basis, n.initial = 6L)
```

---

metric.graph.heat.extend.lower

*Propose a Guarded Lower Graph Heat Time*


---

### Description

Proposes one lower positive heat time after an external search controller has classified the current lower endpoint. The helper enforces a geometric expansion step, an identity-departure floor, a maximum number of expansion rounds, and the truncated-basis resolution certificate. It does not decide whether an endpoint is competitive; fold-level or cohort-level search logic remains the caller's responsibility.

### Usage

```
metric.graph.heat.extend.lower(
    basis,
    eta.grid,
    endpoint.status = c("active", "inactive", "unresolved"),
    expansion.factor = 3,
    max.expansions = 3L,
    expansions.completed = 0L,
    identity.departure = 0.01,
    truncation.tol = 1e-04,
    unresolved.action = c("error", "mark")
)
```

### Arguments

|                      |                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| basis                | A "metric.graph.lowpass.basis" object.                                                                                                                                                                                                                                     |
| eta.grid             | Current finite positive heat-time grid. The exact identity time $\eta = 0$ is intentionally excluded from competitive search.                                                                                                                                              |
| endpoint.status      | External lower-endpoint classification: "active" proposes one extension, while "inactive" and "unresolved" return the grid unchanged.                                                                                                                                      |
| expansion.factor     | Finite numeric factor greater than one. An active proposal is $\min(\eta.grid) / \text{expansion.factor}$ before applying the identity floor.                                                                                                                              |
| max.expansions       | Nonnegative integer maximum number of lower-extension rounds.                                                                                                                                                                                                              |
| expansions.completed | Nonnegative integer number of extension rounds already admitted under the same search contract.                                                                                                                                                                            |
| identity.departure   | Positive number smaller than one. The proposed time cannot be smaller than $-\log(1 - \text{identity.departure}) / \lambda_{\max}$ , where $\lambda_{\max}$ is the largest retained eigenvalue. For a complete basis this is the exact largest graph-Laplacian eigenvalue. |

truncation.tol Positive tolerance smaller than one for the conservative omitted-mode attenuation bound.

unresolved.action Action when an active proposal is not certified by a truncated basis: "error" or "mark".

### Value

A list of class "metric.graph.heat.lower.extension" with the augmented or unchanged grid and proposal telemetry. A proposal is added only when admitted is TRUE.

### Examples

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
lengths <- list(1, c(1, 1), c(1, 1), 1)
basis <- metric.graph.lowpass.basis(adj, lengths, n.eigenpairs = 4L,
                                   eigen.solver = "dense")
eta <- metric.graph.heat.eta.grid(basis, n.initial = 6L)
metric.graph.heat.extend.lower(basis, eta, endpoint.status = "active")
```

---

metric.graph.lowpass.basis

*Construct a Reusable Metric Graph Low-Pass Spectral Basis*

---

### Description

Builds a metric-conductance graph Laplacian and computes the low-frequency eigensystem used by graph low-pass filters. The resulting object is response-independent and can be reused for many responses and filter parameters.

### Usage

```
metric.graph.lowpass.basis(
  adj.list,
  weight.list,
  conductance.rule = c("inverse.length.power", "exp.length", "exp.length.squared",
                       "self.tuned.gaussian"),
  conductance.epsilon = 1e-08,
  conductance.alpha = 1,
  conductance.sigma = NULL,
  conductance.sigma.rule = c("edge.quantile", "median", "local.k"),
  conductance.sigma.quantile = 0.75,
  conductance.local.k = 5L,
  laplacian.type = c("unnormalized", "symmetric.normalized"),
  n.eigenpairs = 50L,
  eigen.solver = c("auto", "sparse", "dense"),
  dense.eigen.threshold = 200L,
  dense.fallback.threshold = 5000L,
```

```

    dense.fallback = c("auto", "never", "always"),
    verbose = FALSE
)

```

### Arguments

|                                         |                                                                                                                                                           |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list</code>                   | List of integer neighbor vectors using 1-based vertex indices.                                                                                            |
| <code>weight.list</code>                | List of positive metric edge lengths parallel to <code>adj.list</code> . These are interpreted as edge lengths, not conductances.                         |
| <code>conductance.rule</code>           | Character scalar. One of "inverse.length.power", "exp.length", "exp.length.squared", or "self.tuned.gaussian".                                            |
| <code>conductance.epsilon</code>        | Positive numeric regularizer used by inverse-power conductances and as a local-scale floor.                                                               |
| <code>conductance.alpha</code>          | Positive numeric exponent for "inverse.length.power".                                                                                                     |
| <code>conductance.sigma</code>          | Optional positive global scale for exponential rules. If NULL, it is selected by <code>conductance.sigma.rule</code> .                                    |
| <code>conductance.sigma.rule</code>     | Rule for selecting a global scale when needed.                                                                                                            |
| <code>conductance.sigma.quantile</code> | Quantile used when <code>conductance.sigma.rule</code> = "edge.quantile".                                                                                 |
| <code>conductance.local.k</code>        | Positive integer local incident-edge order statistic for "self.tuned.gaussian".                                                                           |
| <code>laplacian.type</code>             | Laplacian operator. "unnormalized" uses the weighted graph Laplacian $L = D - C$ . "symmetric.normalized" uses $L_{\text{sym}} = I - D^{-1/2}CD^{-1/2}$ . |
| <code>n.eigenpairs</code>               | Positive integer number of eigenpairs to compute.                                                                                                         |
| <code>eigen.solver</code>               | "auto", "sparse", or "dense".                                                                                                                             |
| <code>dense.eigen.threshold</code>      | Exact dense threshold for auto mode.                                                                                                                      |
| <code>dense.fallback.threshold</code>   | Maximum graph size for emergency dense fallback when sparse decomposition fails and fallback is allowed.                                                  |
| <code>dense.fallback</code>             | "auto", "never", or "always".                                                                                                                             |
| <code>verbose</code>                    | Logical. Reserved for future diagnostic messages.                                                                                                         |

### Details

The basis is complete only when it contains one eigenvector per graph vertex. A truncated basis represents only the retained low-frequency subspace. The largest retained eigenvalue supplies a conservative proxy for bounding the contribution of omitted modes because all omitted eigenvalues are at least as large.

**Value**

A list of class "metric.graph.lowpass.basis" containing the graph operator, eigenvalues, eigenvectors, solver metadata, and spectral completeness diagnostics.

**Examples**

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
lengths <- list(1, c(1, 1), c(1, 1), 1)
metric.graph.lowpass.basis(adj, lengths, n.eigenpairs = 4L,
                           eigen.solver = "dense")
```

---

```
metric.graph.lowpass.operator
```

*Construct a Metric-Conductance Graph Low-Pass Operator*

---

**Description**

Builds a weighted graph Laplacian by transforming metric edge lengths into conductances. This operator is a direct metric-conductance comparator to the legacy rdgraph regression smoother, not a replacement for the Riemannian-complex/overlap-density smoother.

**Usage**

```
metric.graph.lowpass.operator(
  adj.list,
  weight.list,
  conductance.rule = c("inverse.length.power", "exp.length", "exp.length.squared",
    "self.tuned.gaussian"),
  conductance.epsilon = 1e-08,
  conductance.alpha = 1,
  conductance.sigma = NULL,
  conductance.sigma.rule = c("edge.quantile", "median", "local.k"),
  conductance.sigma.quantile = 0.75,
  conductance.local.k = 5L,
  laplacian.type = c("unnormalized", "symmetric.normalized"),
  return.sparse = TRUE,
  verbose = FALSE
)
```

**Arguments**

|                  |                                                                                                                     |
|------------------|---------------------------------------------------------------------------------------------------------------------|
| adj.list         | List of integer neighbor vectors using 1-based vertex indices.                                                      |
| weight.list      | List of positive metric edge lengths parallel to adj.list. These are interpreted as edge lengths, not conductances. |
| conductance.rule | Character scalar. One of "inverse.length.power", "exp.length", "exp.length.squared", or "self.tuned.gaussian".      |

|                            |                                                                                                                                                           |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| conductance.epsilon        | Positive numeric regularizer used by inverse-power conductances and as a local-scale floor.                                                               |
| conductance.alpha          | Positive numeric exponent for "inverse.length.power".                                                                                                     |
| conductance.sigma          | Optional positive global scale for exponential rules. If NULL, it is selected by conductance.sigma.rule.                                                  |
| conductance.sigma.rule     | Rule for selecting a global scale when needed.                                                                                                            |
| conductance.sigma.quantile | Quantile used when conductance.sigma.rule = "edge.quantile".                                                                                              |
| conductance.local.k        | Positive integer local incident-edge order statistic for "self.tuned.gaussian".                                                                           |
| laplacian.type             | Laplacian operator. "unnormalized" uses the weighted graph Laplacian $L = D - C$ . "symmetric.normalized" uses $L_{\text{sym}} = I - D^{-1/2}CD^{-1/2}$ . |
| return.sparse              | Logical. If TRUE, attach a <b>Matrix</b> sparse Laplacian when <b>Matrix</b> is available.                                                                |
| verbose                    | Logical. Reserved for future diagnostic messages.                                                                                                         |

## Details

The legacy rdgraph regression precomputed-graph path uses supplied weight.list values as edge lengths for neighborhood ordering. The spectral conductance in that smoother is overlap-density based:

$$c_e^o = 1 / \max(\rho_1(e), 10^{-10}),$$

where  $\rho_1(e)$  is computed by the Riemannian-complex density machinery.

This function instead constructs conductances directly from metric lengths:

$$c_{ij} = \phi(\ell_{ij}).$$

Supported transforms are

$$c_{ij} = (\ell_{ij} + \epsilon)^{-\alpha},$$

$$c_{ij} = \exp(-\ell_{ij}/\sigma),$$

$$c_{ij} = \exp(-\ell_{ij}^2/\sigma^2),$$

and

$$c_{ij} = \exp(-\ell_{ij}^2/(\sigma_i\sigma_j)).$$

For laplacian.type = "symmetric.normalized", smoothing is performed in the Euclidean eigenbasis of  $L_{\text{sym}}$ . The null vector is proportional to  $\sqrt{d}$ , not the constant vector, so this mode does not preserve constant responses in the same way as the unnormalized Laplacian.

## Value

A list of class "metric.graph.lowpass.operator" containing the edge table, conductances, degree vector, Laplacian triplets, summaries, and optionally a sparse Laplacian matrix.

**Examples**

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
lengths <- list(1, c(1, 1), c(1, 1), 1)
metric.graph.lowpass.operator(adj, lengths)
```

---

model-summary

---

*Summarize a Geometric Smoother or Probability-Mass Fit*


---

**Description**

Summaries expose stored settings and diagnostics without recomputing geometry, fitting, or selection. Console output is bounded; the summary's structured fields retain selected controls, criterion tables, local diagnostic counts, solver information, and density accounting for closer inspection.

**Usage**

```
## S3 method for class 'summary.geosmooth_fit'
print(x, ...)

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

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

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

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

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

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

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

## S3 method for class 'metric.graph.lowpass.fit'
summary(object, ...)

## S3 method for class 'metric.graph.lowpass.refit'
summary(object, ...)

## S3 method for class 'ssrhe.hessian.fit'
```

```

summary(object, ...)

## S3 method for class 'ssrhe.hessian.refit'
summary(object, ...)

## S3 method for class 'ssrhe.hessian.l1.fit'
summary(object, ...)

## S3 method for class 'graph.trend.filtering.fit'
summary(object, ...)

## S3 method for class 'pttf.trend.filtering.fit'
summary(object, ...)

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

## S3 method for class 'lpl_tf'
print(x, ...)

## S3 method for class 'slpl_tf'
print(x, ...)

## S3 method for class 'density_fit'
print(x, ...)

## S3 method for class 'ps_lps'
print(x, ...)

## S3 method for class 'chart_kernel'
print(x, ...)

## S3 method for class 'local_likelihood'
print(x, ...)

```

## Arguments

|           |                                                     |
|-----------|-----------------------------------------------------|
| ...       | Reserved; unused arguments are rejected.            |
| object, x | A fitted object, or a summary for its print method. |

## Details

A finite result is not a convergence certificate. Where the algorithm records no stopping certificate, convergence is explicitly unavailable. Local fallback counts and failed evaluations are reported separately. Selection criteria are conditional tuning scores; they are not an independent assessment of prediction error. GCV, CV MSE, RMSE, and likelihood losses have different interpretations.

**Value**

`summary()` returns a `summary.geosmooth_fit` list with `model`, `n.training`, `n.evaluation`, `n.responses`, `n.observed`, `nonfinite`, `status`, `convergence`, `prediction.domain`, `selected`, `criteria`, `diagnostics`, `solver`, `mass`, `accounting`, and `warnings` fields. Print methods return their input invisibly.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.chart.kernel(X, X[, 1]^2, support.size = 8L)
fit
summary(fit)
mass <- normalize.density(c(1, 2, 3))
mass
summary(mass)$accounting
```

---

model-values

---

*Extract Stored Fitted Values and Response Residuals*


---

**Description**

Use `fitted(object)` for the stored response-scale fitted values, preserving vector or matrix shape, row order, names, and missing values. It does not recompute a fit or imply prediction at new points. For `harmonic.smoother()` results it returns `harmonic_predictions`. The basic `perform.harmonic.smoothing()` result remains an unclassed list, accessed through `$harmonic_predictions`; density probability masses remain `$rho`.

**Usage**

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

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

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

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

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

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

## S3 method for class 'chart_kernel'
```

```
fitted(object, ...)

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

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

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

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

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

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

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

## S3 method for class 'metric.graph.lowpass.fit'
fitted(object, ...)

## S3 method for class 'metric.graph.lowpass.fit'
residuals(object, ...)

## S3 method for class 'metric.graph.lowpass.refit'
fitted(object, ...)

## S3 method for class 'metric.graph.lowpass.refit'
residuals(object, ...)

## S3 method for class 'ssrhe.hessian.fit'
fitted(object, ...)

## S3 method for class 'ssrhe.hessian.fit'
residuals(object, ...)

## S3 method for class 'ssrhe.hessian.refit'
fitted(object, ...)

## S3 method for class 'ssrhe.hessian.refit'
residuals(object, ...)

## S3 method for class 'ssrhe.hessian.l1.fit'
```

```

fitted(object, ...)

## S3 method for class 'ssrhe.hessian.l1.fit'
residuals(object, ...)

## S3 method for class 'graph.trend.filtering.fit'
fitted(object, ...)

## S3 method for class 'graph.trend.filtering.fit'
residuals(object, ...)

## S3 method for class 'pttf.trend.filtering.fit'
fitted(object, ...)

## S3 method for class 'pttf.trend.filtering.fit'
residuals(object, ...)

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

```

### Arguments

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>object</code> | A supported fitted model; see Methods below. |
| <code>...</code>    | Reserved; unused arguments are rejected.     |

### Details

`residuals(object)` returns stored response residuals where available, including missing-label masks. Otherwise it computes  $y - \text{fitted.values}$  only when the evaluation coordinates equal the training coordinates. Residuals at different evaluation points are undefined and cause an error, even when training and evaluation have the same number of rows. Binomial residuals are on the response (probability) scale, not Pearson or deviance residuals. These accessors do not change which refit results are reusable.

### Value

The stored vector or matrix, or response residuals of the same shape.

### See Also

[refit](#), [smoother.matrix](#)

### Examples

```

X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
model <- fit.malps(X, X[, 1]^2, degree = 1L,
  support.type = "knn", support.size = 8L)
head(fitted(model))
head(residuals(model))

```

---

|                   |                                               |
|-------------------|-----------------------------------------------|
| normalize.density | <i>Normalize Fitted Values Into a Density</i> |
|-------------------|-----------------------------------------------|

---

### Description

Converts a numeric field or fitted smoother/regression object into a probability mass vector over the evaluation support. This is the explicit adapter between ordinary smoothers such as [fit.lps](#), [fit.ps.lps](#), and [fit.metric.graph.lowpass](#) and density or occupation-density workflows.

### Usage

```
normalize.density(x, ...)

## S3 method for class 'numeric'
normalize.density(
  x,
  X = NULL,
  density.control = list(),
  method.id = "normalized_numeric",
  keep.source.fit = FALSE,
  adj.list = NULL,
  empirical.rho = NULL,
  return.details = TRUE,
  ...
)

## Default S3 method:
normalize.density(
  x,
  X = NULL,
  density.control = list(),
  method.id = NULL,
  keep.source.fit = TRUE,
  adj.list = NULL,
  empirical.rho = NULL,
  return.details = TRUE,
  ...
)

## S3 method for class 'lps'
normalize.density(
  x,
  X = NULL,
  density.control = list(),
  method.id = NULL,
  keep.source.fit = TRUE,
  adj.list = NULL,
```

```

    empirical.rho = NULL,
    return.details = TRUE,
    ...
)

## S3 method for class 'ps_lps'
normalize.density(
  x,
  X = NULL,
  density.control = list(),
  method.id = NULL,
  keep.source.fit = TRUE,
  adj.list = NULL,
  empirical.rho = NULL,
  return.details = TRUE,
  ...
)

## S3 method for class 'metric.graph.lowpass.fit'
normalize.density(
  x,
  X = NULL,
  density.control = list(),
  method.id = NULL,
  keep.source.fit = TRUE,
  adj.list = NULL,
  empirical.rho = NULL,
  return.details = TRUE,
  ...
)

## S3 method for class 'metric.graph.lowpass.refit'
normalize.density(
  x,
  X = NULL,
  density.control = list(),
  method.id = NULL,
  keep.source.fit = TRUE,
  adj.list = NULL,
  empirical.rho = NULL,
  return.details = TRUE,
  ...
)

```

### Arguments

|     |                                                             |
|-----|-------------------------------------------------------------|
| x   | Numeric vector or fitted object with a fitted.values field. |
| ... | Additional arguments passed to methods.                     |

|                              |                                                                                                                                                                                                                        |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>               | Optional support/evaluation matrix. If omitted, methods use the fitted object's stored <code>X.eval</code> or <code>X</code> field when available; for a bare numeric vector, a one-dimensional index support is used. |
| <code>density.control</code> | List controlling clipping, normalization, and accounting checks. See <a href="#">fit.density</a> .                                                                                                                     |
| <code>method.id</code>       | Character method identifier recorded in the returned object.                                                                                                                                                           |
| <code>keep.source.fit</code> | Logical; if TRUE, retain the source fit in diagnostics for object methods.                                                                                                                                             |
| <code>adj.list</code>        | Optional adjacency list used to compute graph-local smoothness diagnostics for the normalized density.                                                                                                                 |
| <code>empirical.rho</code>   | Optional empirical probability mass vector used for accounting diagnostics.                                                                                                                                            |
| <code>return.details</code>  | Logical; if TRUE, keep diagnostic details in the result.                                                                                                                                                               |

**Value**

A list of class "density\_fit".

**Examples**

```
normalize.density(c(0.5, -0.1, 0.6, 0), X = matrix(1:4, ncol = 1))
```

---

```
perform.harmonic.smoothing
```

*Perform Harmonic Smoothing on Graph Function Values*

---

**Description**

Applies harmonic smoothing to function values defined on vertices of a graph, preserving values at the boundary of a specified region while smoothly interpolating interior values. This function implements a discrete Laplace equation solution using weighted averaging.

**Usage**

```
perform.harmonic.smoothing(
  adj.list,
  weight.list,
  values,
  region.vertices,
  max.iterations = 100,
  tolerance = 1e-06
)
```

**Arguments**

|                              |                                                                                                                                          |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list</code>        | A list of integer vectors, where each vector contains indices of vertices adjacent to the corresponding vertex. Indices must be 1-based. |
| <code>weight.list</code>     | A list of numeric vectors containing weights of edges corresponding to adjacencies in <code>adj.list</code> .                            |
| <code>values</code>          | A numeric vector of function values defined at each vertex.                                                                              |
| <code>region.vertices</code> | An integer vector of vertex indices (1-based) defining the region to be smoothed. Boundary vertices will have fixed values.              |
| <code>max.iterations</code>  | Integer scalar, the maximum number of relaxation iterations to perform. Default is 100.                                                  |
| <code>tolerance</code>       | Numeric scalar, the convergence threshold for value changes. Default is 1e-6.                                                            |

**Details**

Boundary vertices are exactly the region vertices with a neighbor outside the region. Degree-one vertices are not automatically boundary vertices. Boundary values and values outside the region remain fixed. Each synchronous relaxation step replaces interior values by neighbor averages with conductance  $1/(w + 10^{-10})$ , where  $w$  is the supplied edge length.

With no boundary, this function returns the input unchanged, with status "no\_boundary" and converged = FALSE. With no interior there is nothing to solve: status is "no\_interior" and convergence is true. On a disconnected region, components without a connection to fixed boundary values are not uniquely anchored; relaxation can oscillate. Inspect convergence before using the result as a harmonic solution. Unlike this basic function, `harmonic.smoother()` also runs relaxation when the whole region has no boundary, retaining its historical behavior.

**Value**

A list containing:

- `harmonic_predictions`: the numeric vector of smoothed values;
- `converged`: whether both the maximum update and discrete harmonic residual fell below tolerance;
- `num_region`, `num_boundary`, and `num_interior`: vertex counts for the requested region and its boundary/interior split;
- `num_iterations`: the number of relaxation iterations run;
- `max_change` and `max_residual`: per-iteration convergence diagnostics;
- `status`: converged, iteration\_limit, no\_boundary, or no\_interior.

**See Also**

[harmonic.smoother](#) for smoothing with topology tracking, [get.region.boundary](#) for boundary vertex identification

**Examples**

```
adj.list <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
weight.list <- list(1, c(1, 1), c(1, 1), 1)
values <- c(0, 2, -1, 1)
result <- perform.harmonic.smoothing(
  adj.list, weight.list, values, region.vertices = 1:3
)
result$harmonic_predictions
```

---

plot.harmonic\_smoother

*Plot Method for Harmonic Smoother Results*


---

**Description**

Creates various plots to visualize the results of harmonic smoothing with topology tracking.

**Usage**

```
## S3 method for class 'harmonic_smoother'
plot(x, y = NULL, ..., type = c("topology", "extrema", "values"))
```

**Arguments**

|      |                                                                                                                                                                                                                            |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | An object of class "harmonic_smoother".                                                                                                                                                                                    |
| y    | Ignored (included for S3 method consistency).                                                                                                                                                                              |
| ...  | Additional graphical parameters passed to plot().                                                                                                                                                                          |
| type | Character string specifying the type of plot. Options are:<br><b>"topology"</b> Evolution of topology differences (default)<br><b>"extrema"</b> Evolution of extrema counts<br><b>"values"</b> Original vs smoothed values |

**Value**

Invisibly returns the input object.

**See Also**

[harmonic.smoother](#)

**Examples**

```
adj.list <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
weight.list <- list(1, c(1, 1), c(1, 1), 1)
values <- c(0, 2, -1, 1)
result <- harmonic.smoother(
  adj.list, weight.list, values, region.vertices = seq_along(values),
  max.iterations = 20, record.frequency = 2
)
plot(result, type = "values")
```

---

plot.synthetic\_dataset

*Plot a canonical synthetic dataset*


---

**Description**

The default view = "auto" preserves the dimension-dependent display: one latent coordinate shows response and truth against that coordinate; otherwise the first two predictor coordinates show a geometry projection. Select "geometry" explicitly to see a curve's embedding, or "response" to see responses against the first latent coordinate (observation index when latent coordinates are absent). A two-coordinate projection can hide separation in higher ambient dimensions. Plotting does not draw random values.

**Usage**

```
## S3 method for class 'synthetic_dataset'
plot(
  x,
  color = c("response", "truth", "region"),
  ...,
  view = c("auto", "geometry", "response"),
  legend = TRUE,
  col = NULL
)
```

**Arguments**

|        |                                                                                                                                            |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------|
| x      | A synthetic_dataset.                                                                                                                       |
| color  | Color mapping for points: response, truth, or region. Region mapping requires stored region labels. Numeric keys display the actual range. |
| ...    | Additional arguments passed to <code>graphics::plot()</code> , including overrides for axis labels, point shape, or color.                 |
| view   | "auto", "geometry", or "response". Geometry requires at least two predictor coordinates. Named-only, after ..., for compatibility.         |
| legend | Logical; show point-color and response/truth keys. A user-supplied col suppresses the automatic color key.                                 |
| col    | Optional explicit point color, overriding the color mapping.                                                                               |

**Value**

x, invisibly.

**Examples**

```
x <- materialize.synthetic(synthetic.registry.spec("G1"), n = 20L, seed = 1L)
plot(x)
plot(x, view = "geometry", color = "truth")
```

---

predict.lpl\_tf

*Predict From A Local Polynomial Lifting Trend Filter*

---

**Description**

Predict From A Local Polynomial Lifting Trend Filter

**Usage**

```
## S3 method for class 'lpl_tf'
predict(
  object,
  newdata = NULL,
  type = c("response"),
  allow.incomplete = FALSE,
  ...
)
```

**Arguments**

|                  |                                                                              |
|------------------|------------------------------------------------------------------------------|
| object           | A "lpl_tf" object.                                                           |
| newdata          | Must be NULL; prediction is currently available only at the training points. |
| type             | Prediction type. The current implementation supports only "response".        |
| allow.incomplete | Reserved.                                                                    |
| ...              | Reserved.                                                                    |

**Value**

Fitted values at the training points.

**Examples**

```
if (requireNamespace("genlasso", quietly = TRUE)) {
  X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
  fit <- fit.lpl.tf(X, sin(2 * pi * X[, 1]), degree = 1L,
    support.type = "knn", support.size = 7L,
    lambda = 0.1, lambda.selection = "fixed")
  predict(fit)
}
```

---

|             |                                |
|-------------|--------------------------------|
| predict.lps | <i>Predict from an LPS Fit</i> |
|-------------|--------------------------------|

---

**Description**

Predict from an LPS Fit

**Usage**

```
## S3 method for class 'lps'
predict(object, newdata = NULL, type = c("response", "raw"), ...)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                             |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object  | A fitted "lps" object.                                                                                                                                                                                                                                                                      |
| newdata | Optional prediction matrix. Defaults to the fitted object's stored <code>X.eval</code> .                                                                                                                                                                                                    |
| type    | Prediction scale. "response" returns response-scale predictions; for <code>outcome.family = "bernoulli"</code> or "binomial" these are probabilities in $[0,1]$ . "raw" returns the unmodified local least-squares predictions for "bernoulli" and the fitted probabilities for "binomial". |
| ...     | Unused.                                                                                                                                                                                                                                                                                     |

**Value**

A numeric vector of predictions.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
fit <- fit.lps(X, sin(2 * pi * X[, 1]), support.grid = 6L)
predict(fit)
```

---

|               |                                 |
|---------------|---------------------------------|
| predict.malps | <i>Predict From A MALPS Fit</i> |
|---------------|---------------------------------|

---

**Description**

Predict From A MALPS Fit

**Usage**

```
## S3 method for class 'malps'
predict(
  object,
  newdata = NULL,
  type = c("response"),
  allow.incomplete = FALSE,
  ...
)
```

**Arguments**

|                  |                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object           | A "malps" object from <a href="#">fit.malps</a> .                                                                                                                                                                                                                                                                                                                               |
| newdata          | Optional numeric coordinate matrix with the same number of columns as object\$X. If NULL, returns stored training-point fitted values. New-point prediction is currently implemented for coordinate-support fits only; it fails informatively for objects fitted with support.metric = "graph.geodesic" because new targets need a graph-attachment/geodesic-distance contract. |
| type             | Prediction type. Currently only "response".                                                                                                                                                                                                                                                                                                                                     |
| allow.incomplete | Logical; if FALSE, new-point prediction fails when any target has no positive averaging coverage. If TRUE, those targets receive NA_real_.                                                                                                                                                                                                                                      |
| ...              | Reserved for future extensions.                                                                                                                                                                                                                                                                                                                                                 |

**Value**

Numeric vector of fitted values.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
fit <- fit.malps(X, sin(2 * pi * X[, 1]), support.size = 6L)
predict(fit)
```

---

predict.slpl\_tf

*Predict From A Fixed-Operator Synchronized LPL-TF Model*

---

**Description**

Predict From A Fixed-Operator Synchronized LPL-TF Model

**Usage**

```
## S3 method for class 'slpl_tf'
predict(object, newdata = NULL, type = c("response"), ...)
```

**Arguments**

|         |                                                                              |
|---------|------------------------------------------------------------------------------|
| object  | A "slpl_tf" fit.                                                             |
| newdata | Must be NULL; prediction is currently available only at the training points. |
| type    | Prediction type. The current implementation supports only "response".        |
| ...     | Reserved.                                                                    |

**Value**

Fitted values at the training points.

**Examples**

```
if (requireNamespace("genlasso", quietly = TRUE)) {
  X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
  fit <- fit.slpl.tf(X, sin(2 * pi * X[, 1]), degree = 1L,
                    support.type = "knn", support.size = 7L,
                    lambda1 = 0.1, lambda2 = 0,
                    lambda.selection = "fixed")
  predict(fit)
}
```

---

```
print.graph.trend.filtering.fit
```

*Print Geosmooth S3 Objects*

---

**Description**

Print concise summaries for fitted geosmooth models and constructed operators.

**Usage**

```
## S3 method for class 'graph.trend.filtering.fit'
print(x, ...)

## S3 method for class 'lpl_tf_operator'
print(x, ...)

## S3 method for class 'lps'
print(x, ...)

## S3 method for class 'malps'
print(x, ...)

## S3 method for class 'malps_bootstrap'
print(x, ...)
```

```
## S3 method for class 'metric.graph.lowpass.fit'
print(x, ...)

## S3 method for class 'metric.graph.lowpass.refit'
print(x, ...)

## S3 method for class 'pttf.trend.filtering.fit'
print(x, ...)

## S3 method for class 'pttf_operator'
print(x, ...)

## S3 method for class 'slpl_tf_operator'
print(x, ...)

## S3 method for class 'ssrhe.hessian.operator'
print(x, ...)

## S3 method for class 'ssrhe.hessian.l1.fit'
print(x, ...)

## S3 method for class 'ssrhe.hessian.fit'
print(x, ...)

## S3 method for class 'ssrhe.hessian.refit'
print(x, ...)

## S3 method for class 'ssrhe.hessian.cv.fit'
print(x, ...)

## S3 method for class 'ssrhe.hessian.gcv.fit'
print(x, ...)

## S3 method for class 'transported.graph.hessian.operator'
print(x, ...)
```

### Arguments

|                  |                                          |
|------------------|------------------------------------------|
| <code>x</code>   | A geosmooth S3 object.                   |
| <code>...</code> | Additional arguments, currently ignored. |

### Value

Invisibly returns `x`.

### Examples

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
fit <- fit.lps(X, sin(2 * pi * X[, 1]), support.grid = 6L)
```

```
print(fit)
```

---

```
print.harmonic_smoother
```

*Print Method for Harmonic Smoother Results*

---

## Description

Prints a concise summary of harmonic smoother results.

## Usage

```
## S3 method for class 'harmonic_smoother'  
print(x, ...)
```

## Arguments

|     |                                                    |
|-----|----------------------------------------------------|
| x   | An object of class "harmonic_smoother".            |
| ... | Further arguments passed to or from other methods. |

## Value

Invisibly returns the input object.

## See Also

[harmonic.smoother](#), [summary.harmonic\\_smoother](#)

## Examples

```
fit <- structure(  
  list(  
    stable_iteration = 2L,  
    i_harmonic_predictions = matrix(1:4, nrow = 2),  
    i_basins = list(matrix(c(1, 1), ncol = 2)),  
    topology_differences = numeric()  
  ),  
  class = "harmonic_smoother"  
)  
print(fit)
```

---

```
print.summary.harmonic_smoother
```

*Print Method for Summary of Harmonic Smoother Results*

---

## Description

Prints the summary of harmonic smoother results in a formatted manner.

## Usage

```
## S3 method for class 'summary.harmonic_smoother'  
print(x, ...)
```

## Arguments

|     |                                                    |
|-----|----------------------------------------------------|
| x   | An object of class "summary.harmonic_smoother".    |
| ... | Further arguments passed to or from other methods. |

## Value

Invisibly returns the input object.

## See Also

[summary.harmonic\\_smoother](#)

## Examples

```
object <- structure(  
  list(  
    stable_iteration = 2L,  
    iterations_recorded = 1L,  
    initial_extrema = 1L,  
    initial_maxima = 1L,  
    initial_minima = 0L,  
    final_extrema = 1L,  
    final_maxima = 1L,  
    final_minima = 0L,  
    extrema_reduction = 0L,  
    topology_diff_summary = NULL  
  ),  
  class = "summary.harmonic_smoother"  
)  
print(object)
```

---

```
print.synthetic_dataset
```

*Print a canonical synthetic dataset*

---

**Description**

Print a canonical synthetic dataset

**Usage**

```
## S3 method for class 'synthetic_dataset'
print(x, ...)
```

**Arguments**

|     |                      |
|-----|----------------------|
| x   | A synthetic_dataset. |
| ... | Ignored.             |

**Value**

x, invisibly.

**Examples**

```
x <- materialize.synthetic(synthetic.registry.spec("G1"), n = 20L, seed = 1L)
print(x)
```

---

```
pttf.geometry
```

*Construct PTF Local Geometry*

---

**Description**

Builds the geometry layer for the experimental parallel-transport trend filtering (PTTF) route. The returned object contains graph supports, local PCA tangent frames, edge directions, orthogonal edge transports, sampling density metadata, and diagnostics. It does not assemble transported difference operators and it does not fit a response.

**Usage**

```
pttf.geometry(
  X,
  adj.list = NULL,
  weight.list = NULL,
  graph = c("rknn", "supplied"),
  tangent.dim,
  local.support = c("graph.disk", "neighbors"),
```

```

min.support = NULL,
max.hops = 3L,
support.k = NULL,
graph.k.scale = 1L,
graph.radius.factor = 1,
graph.radius.rule = "geomean",
transport.rule = c("procrustes"),
synchronize.orientation = TRUE,
density.method = c("support.radius"),
alpha = 1,
diagnostics = TRUE
)

```

### Arguments

|                                                                                                |                                                                                                                                                                                     |
|------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                                                                                 | Numeric data matrix with one row per vertex.                                                                                                                                        |
| <code>adj.list</code>                                                                          | Optional supplied graph adjacency list using 1-based vertex indices.                                                                                                                |
| <code>weight.list</code>                                                                       | Optional positive edge-length list parallel to <code>adj.list</code> .                                                                                                              |
| <code>graph</code>                                                                             | Graph source. "rknn" builds an adaptive-radius graph from <code>X</code> ; "supplied" uses <code>adj.list</code> and <code>weight.list</code> .                                     |
| <code>tangent.dim</code>                                                                       | Fixed global tangent dimension for local geometry.                                                                                                                                  |
| <code>local.support</code>                                                                     | Local support construction rule. "graph.disk" uses graph-hop disks; "neighbors" starts from the closed one-hop neighborhood and tops up by graph-hop expansion.                     |
| <code>min.support</code>                                                                       | Minimum support size. Defaults to $\max(2m+2, m+3)$ , where $m = \text{tangent.dim}$ .                                                                                              |
| <code>max.hops</code>                                                                          | Maximum hop radius used when topping up local supports.                                                                                                                             |
| <code>support.k</code>                                                                         | Optional cap on local support size after top-up. The center vertex is always retained and remaining vertices are selected by graph metric distance, with vertex-index tie breaking. |
| <code>graph.k.scale</code> , <code>graph.radius.factor</code> , <code>graph.radius.rule</code> | Adaptive-radius graph controls passed to <code>dgraphs::create.rknn.graph()</code> when <code>graph = "rknn"</code> .                                                               |
| <code>transport.rule</code>                                                                    | Edge transport rule. The current implementation uses "procrustes".                                                                                                                  |
| <code>synchronize.orientation</code>                                                           | Logical. If TRUE, return deterministic spanning-tree orientation-synchronization metadata. This metadata does not mutate the returned frames.                                       |
| <code>density.method</code>                                                                    | Sampling-density metadata rule. The current implementation uses "support.radius".                                                                                                   |
| <code>alpha</code>                                                                             | Numeric exponent used only to report proposed future density-normalization weights.                                                                                                 |
| <code>diagnostics</code>                                                                       | Logical. If TRUE, compute cycle diagnostics.                                                                                                                                        |

### Details

The stored transport convention is column-vector based. For an oriented edge  $i \leftarrow j$ ,  $O_{ij}$  maps tangent coordinates from frame  $j$  into frame  $i$ :

$$v_j \mapsto O_{ij}v_j.$$

For shared-support Procrustes, row-coordinate matrices  $Z_i$  and  $Z_j$  are formed on the same vertex set. The row problem  $\min_Q \|Z_j Q - Z_i\|_F^2$  gives  $Q = UV^\top$  when  $Z_j^\top Z_i = U \Sigma V^\top$ . The stored column map is  $O_{ij} = Q^\top = VU^\top$ ; the reported residual is  $\|Z_j O_{ij}^\top - Z_i\|_F / \max(\|Z_i\|_F, \epsilon)$ .

### Value

A list of class "pttf\_geometry".

### Examples

```
n <- 10L
X <- matrix(seq(0, 1, length.out = n), ncol = 1)
adj <- lapply(seq_len(n), function(i) intersect(c(i - 1L, i + 1L), 1:n))
lengths <- Map(function(i, j) abs(X[j, 1] - X[i, 1]), seq_len(n), adj)
pttf.geometry(X, adj, lengths, graph = "supplied", tangent.dim = 1L)
```

---

pttf.operator

---

Construct PTTF Transported-Difference Operators

---

### Description

Assembles experimental parallel-transport trend-filtering (PTTF) transported-difference operators from a [pttf.geometry](#) object. This is an operator-construction function only: it does not fit a response, tune a penalty parameter, or run an  $\ell_1/\ell_2$  smoother.

### Usage

```
pttf.operator(
  geometry,
  derivative.order = 3L,
  edge.status.policy = c("ok.only", "frame.fallback"),
  regression.weight.rule = c("inverse.length.squared", "inverse.length", "unit"),
  edge.length.epsilon = 1e-08,
  tensor.scaling = c("hs", "raw"),
  row.mass.rule = c("node.mass", "none"),
  row.normalize = c("none", "l2"),
  min.operator.rank.tol = 1e-10,
  max.operator.condition = 1e+08,
  return.intermediate = TRUE,
  return.A = TRUE,
  return.B = TRUE,
  diagnostics = TRUE
)
```

## Arguments

|                        |                                                                                                                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| geometry               | A "pttf_geometry" object from <a href="#">pttf.geometry</a> .                                                                                                                                 |
| derivative.order       | Integer derivative order, one of 1L, 2L, or 3L.                                                                                                                                               |
| edge.status.policy     | Edge transport policy. "ok.only" uses only transports with status "ok". "frame.fallback" also allows direct-frame fallback transports for non-OK shared-support statuses.                     |
| regression.weight.rule | Local regression edge-weight rule.                                                                                                                                                            |
| edge.length.epsilon    | Positive numeric edge-length floor.                                                                                                                                                           |
| tensor.scaling         | Symmetric tensor coordinate scaling. "hs" uses square-root multiplicity scaling so coordinate Euclidean norms match Hilbert–Schmidt norms. "raw" stores unscaled unique symmetric components. |
| row.mass.rule          | Final row-mass multiplier. "node.mass" multiplies rows for vertex $i$ by $\sqrt{\mu_i}$ from the density metadata. "none" leaves rows unweighted.                                             |
| row.normalize          | Optional final row normalization.                                                                                                                                                             |
| min.operator.rank.tol  | Relative singular-value tolerance for local derivative regression rank.                                                                                                                       |
| max.operator.condition | Maximum allowed weighted local-design condition number.                                                                                                                                       |
| return.intermediate    | Logical. If TRUE, include full logical intermediate derivative matrices.                                                                                                                      |
| return.A               | Logical. If TRUE, include the compact sparse row operator.                                                                                                                                    |
| return.B               | Logical. If TRUE, include $A^\top A$ .                                                                                                                                                        |
| diagnostics            | Logical. If TRUE, include assembly diagnostics.                                                                                                                                               |

## Details

The operator keeps a full logical vertex/component layout internally. For order  $r$ , the logical field has one block of  $q_r = \binom{m+r-1}{r}$  components at each vertex. The returned  $A$  is compact: it includes rows only for accepted final-order vertex/component blocks, and `row.table$full.row` records the original full logical row.

For derivative level  $r$ , the local regression uses blocks  $\Phi_r(u_{ij}) \in \mathbb{R}^{q_{r-1} \times q_r}$ , lifted weights  $W_i^{(r)} = \text{diag}(w_{ij}) \otimes I_{q_{r-1}}$ , and transported predecessor differences  $K_{r-1}(O_{ij})z_j - z_i$ . Unavailable predecessor blocks are never filled with zero; affected neighbors are dropped and recorded.

## Value

A list of class "pttf\_operator" containing sparse operators, row/vertex provenance, tensor-basis metadata, and diagnostics.

**Examples**

```

n <- 10L
X <- matrix(seq(0, 1, length.out = n), ncol = 1)
adj <- lapply(seq_len(n), function(i) intersect(c(i - 1L, i + 1L), 1:n))
lengths <- Map(function(i, j) abs(X[j, 1] - X[i, 1]), seq_len(n), adj)
geometry <- pttf.geometry(
  X, adj, lengths, graph = "supplied", tangent.dim = 1L
)
pttf.operator(geometry, derivative.order = 2L)

```

---

pttf.operator.filter.rows

*Filter Rows Of A PTTF Operator*


---

**Description**

Creates a new "pttf\_operator" object with a subset of compact operator rows. The sparse triplet payload is rebuilt from the filtered matrix so triplet row indices always refer to the filtered fit rows, while original compact and full logical row identities remain in row.table.

**Usage**

```

pttf.operator.filter.rows(
  operator,
  rows,
  reason = NULL,
  preserve.original = TRUE
)

```

**Arguments**

|                   |                                                                                                         |
|-------------------|---------------------------------------------------------------------------------------------------------|
| operator          | A "pttf_operator" object from <a href="#">pttf.operator</a> .                                           |
| rows              | Integer, logical, or character row selector. Character selectors are matched against row.table\$status. |
| reason            | Optional character reason stored in row.filter.                                                         |
| preserve.original | Logical. If TRUE, preserve original row identities in row.table\$compact.row and row.table\$full.row.   |

**Value**

A filtered "pttf\_operator" object.

## Examples

```
n <- 10L
X <- matrix(seq(0, 1, length.out = n), ncol = 1)
adj <- lapply(seq_len(n), function(i) intersect(c(i - 1L, i + 1L), 1:n))
lengths <- Map(function(i, j) abs(X[j, 1] - X[i, 1]), seq_len(n), adj)
geometry <- pttf.geometry(
  X, adj, lengths, graph = "supplied", tangent.dim = 1L
)
operator <- pttf.operator(geometry, derivative.order = 2L)
pttf.operator.filter.rows(operator, seq_len(max(1L, nrow(operator$A) - 2L)))
```

---

refit

*Refit A Model Using Its Stored Geometry*


---

## Description

Fit a new response while reusing the geometry or operators in a supported fitted object. R selects the method from the class of object.

## Usage

```
refit(object, y, ...)
```

```
## Default S3 method:
refit(object, y, ...)
```

## Arguments

|                     |                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>object</code> | A fitted MALPS, local polynomial lifting trend filter, synchronized lifting trend filter, metric graph low-pass, quadratic Hessian-energy, or Hessian L1 model.                                                                                                                                                                                                          |
| <code>y</code>      | New response. MALPS and the two lifting trend filters accept a vector; Hessian L1 accepts one response; metric graph low-pass and quadratic Hessian methods also accept a matrix with one response per column. MALPS and both Hessian methods reuse the stored response when <code>y</code> is omitted or <code>NULL</code> . The other methods require <code>y</code> . |
| <code>...</code>    | Method-specific controls, described in the linked method help. Unrecognized arguments are errors.                                                                                                                                                                                                                                                                        |

## Details

Refitting does not automatically repeat the original model selection:

`malps` [refit.malps](#) reuses supports and averaging weights; accepts observation weights. Robust local residual weights, if used, are recomputed for the new response.

`lpl_tf` [refit.lpl\\_tf](#) reuses the residual operator and penalty, or accepts a new fixed `lambda`.

`slpl_tf` [refit.slpl\\_tf](#) reuses both operators and penalties, or accepts fixed `lambda1` and `lambda2`.

`metric.graph.lowpass.fit` [refit.metric.graph.lowpass.fit](#) reuses the spectral basis and smoothing parameter. Set `per.column.gcv = TRUE` to select a parameter for each response column.

`ssrhe.hessian.fit` [refit.ssrhe.hessian.fit](#) reuses the Hessian operator and fixed penalties, with optional `lambda1`, `lambda2`, `weights`, and `ridge`. This also handles CV- and GCV-selected fits, retaining their selected penalties by default.

`ssrhe.hessian.l1.fit` [refit.ssrhe.hessian.l1.fit](#) reuses the derivative operator with fixed or CV-selected penalties; accepts `lambda.grid`, `lambda.selection`, and solver controls.

Use the original fitted object for repeated metric graph low-pass or quadratic Hessian refits: their refit results are summaries, not reusable input fits. Other supported families return reusable fitted objects.

### Value

A refitted object or refit summary, as documented by the method.

### Migration

The six former exported `refit.*()` functions are replaced by `refit(object, y, ...)`. Replace named `fitted.model` and `y.new` arguments with `object` and `y`, respectively. Method-specific controls and return classes are preserved. Call the generic rather than an individual S3 method; the former help names remain available for navigation.

### See Also

[smoother.matrix](#)

### Examples

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.malps(X, X[, 1]^2, degree = 1L,
               support.type = "knn", support.size = 8L)
updated <- refit(fit, y = X[, 1]^3)
head(predict(updated))
```

---

refit.lpl\_tf

---

*Refit A Local Polynomial Lifting Trend Filter*


---

### Description

Reuses the fixed operator and selected `lambda` from an existing LPL-TF fit.

### Usage

```
## S3 method for class 'lpl_tf'
refit(object, y, lambda = NULL, reuse.lambda = TRUE, verbose = FALSE, ...)
```

**Arguments**

|              |                                                           |
|--------------|-----------------------------------------------------------|
| object       | A "lpl_tf" object.                                        |
| y            | New numeric response vector.                              |
| lambda       | Optional fixed lambda.                                    |
| reuse.lambda | Logical. If TRUE and lambda = NULL, reuse object\$lambda. |
| verbose      | Logical.                                                  |
| ...          | Reserved.                                                 |

**Value**

A refitted "lpl\_tf" object.

**See Also**

[refit](#) for supported fitted objects and migration.

**Examples**

```
if (requireNamespace("genlasso", quietly = TRUE)) {
  X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
  fit <- fit.lpl.tf(X, X[, 1]^2, degree = 1L,
    support.type = "knn", support.size = 7L,
    lambda = 0.1, lambda.selection = "fixed")
  refit(fit, y = X[, 1]^3)
}
```

---

 refit.malps

---

*Refit A MALPS Smoother With A New Response*


---

**Description**

Reuses the anchors, selected support profile, local supports, prediction supports, coordinate method, kernel, degree, and local-solver controls from a previous [fit.malps](#) result, then recomputes local polynomial coefficients for a new response vector. Observation weights, when supplied, multiply the stored geometric fitting weights for local coefficient estimation. Supports and model-averaging weights remain fixed. A weighted refit requires every stored local support to retain at least one positive effective fitting weight; sparse bootstrap-style weights can therefore fail even when the global weight vector is not all zero. If the original fit used robust local residual reweighting, robust weights are recomputed for the new response because they are response-dependent.

**Usage**

```
## S3 method for class 'malps'
refit(
  object,
  y = NULL,
  weights = NULL,
  reuse.selection = TRUE,
  refit.local.coefficients = TRUE,
  verbose = FALSE,
  ...
)
```

**Arguments**

|                                       |                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>object</code>                   | A "malps" object from <a href="#">fit.malps</a> .                                                                                                                                                                                                                                                                                                        |
| <code>y</code>                        | Optional new numeric response vector with length <code>nrow(object\$X)</code> . If <code>NULL</code> , the original response is reused.                                                                                                                                                                                                                  |
| <code>weights</code>                  | Optional nonnegative numeric observation weights with length <code>nrow(object\$X)</code> . Zero weights remove observations from local coefficient estimation but do not change the stored supports or averaging weights. The refit fails if any local support has no positive effective fitting weight after multiplying by these observation weights. |
| <code>reuse.selection</code>          | Currently requires <code>TRUE</code> .                                                                                                                                                                                                                                                                                                                   |
| <code>refit.local.coefficients</code> | Currently requires <code>TRUE</code> .                                                                                                                                                                                                                                                                                                                   |
| <code>verbose</code>                  | Logical; reserved for future progress messages.                                                                                                                                                                                                                                                                                                          |
| <code>...</code>                      | Reserved for future extensions.                                                                                                                                                                                                                                                                                                                          |

**Value**

A "malps" object with the same support profile and new fitted values.

**See Also**

[refit](#) for supported fitted objects and migration.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.malps(X, X[, 1]^2, degree = 1L,
  support.type = "knn", support.size = 8L)
refit(fit, y = X[, 1]^3)
```

---

refit.metric.graph.lowpass.fit

*Refit Metric-Conductance Graph Low-Pass Regression*


---

## Description

Reuses a fitted metric graph low-pass eigensystem to smooth new responses.

## Usage

```
## S3 method for class 'metric.graph.lowpass.fit'
refit(
  object,
  y,
  per.column.gcv = FALSE,
  eta.grid = NULL,
  n.candidates = 40L,
  n.cores = 1L,
  block.size = NULL,
  verbose = FALSE,
  ...
)
```

## Arguments

|                |                                                                                                                                                                                                             |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object         | A "metric.graph.lowpass.fit" object.                                                                                                                                                                        |
| y              | Numeric vector or matrix with one row per graph vertex.                                                                                                                                                     |
| per.column.gcv | Logical. If TRUE, select eta independently for each response column using the cached eigenbasis.                                                                                                            |
| eta.grid       | Optional positive numeric eta grid for per-column GCV.                                                                                                                                                      |
| n.candidates   | Number of eta candidates when eta.grid = NULL.                                                                                                                                                              |
| n.cores        | Positive integer retained for compatibility. Response columns are always processed sequentially; values greater than one do not enable parallel execution. Per-column GCV results report n.cores.used = 1L. |
| block.size     | Optional block size for fixed-eta multi-column refits.                                                                                                                                                      |
| verbose        | Logical progress flag.                                                                                                                                                                                      |
| ...            | Additional arguments are not currently supported.                                                                                                                                                           |

## Value

A list of class "metric.graph.lowpass.refit".

## See Also

[refit](#) for supported fitted objects and migration.

**Examples**

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
lengths <- list(1, c(1, 1), c(1, 1), 1)
fit <- fit.metric.graph.lowpass(
  adj, lengths, y = 1:4, n.eigenpairs = 4L,
  eta.grid = c(0.1, 1), eigen.solver = "dense"
)
refit(fit, y = 4:1)
```

refit.slpl\_tf

*Refit A Fixed-Operator Synchronized LPL-TF Model***Description**

Refit A Fixed-Operator Synchronized LPL-TF Model

**Usage**

```
## S3 method for class 'slpl_tf'
refit(
  object,
  y,
  lambda1 = NULL,
  lambda2 = NULL,
  reuse.lambda = TRUE,
  verbose = FALSE,
  ...
)
```

**Arguments**

|                  |                                                                               |
|------------------|-------------------------------------------------------------------------------|
| object           | A "slpl_tf" fit.                                                              |
| y                | New numeric response vector.                                                  |
| lambda1, lambda2 | Optional fixed penalties. Defaults reuse object\$lambda1 and object\$lambda2. |
| reuse.lambda     | Logical. If TRUE, reuse omitted penalties.                                    |
| verbose          | Logical.                                                                      |
| ...              | Reserved.                                                                     |

**Value**

A refitted "slpl\_tf" object.

**See Also**

[refit](#) for supported fitted objects and migration.

**Examples**

```

if (requireNamespace("genlasso", quietly = TRUE)) {
  X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
  fit <- fit.slpl.tf(X, X[, 1]^2, degree = 1L,
                    support.type = "knn", support.size = 7L,
                    lambda1 = 0.1, lambda2 = 0,
                    lambda.selection = "fixed")
  refit(fit, y = X[, 1]^3)
}

```

---

```
refit.ssrhe.hessian.fit
```

*Refit SSRHE-Style Hessian-Energy Regression*

---

**Description**

Reuses the operator from [fit.ssrhe.hessian.regression](#) to fit new responses or new fixed penalty weights without rebuilding local PCA neighborhoods or Hessian-energy matrices.

**Usage**

```

## S3 method for class 'ssrhe.hessian.fit'
refit(
  object,
  y = NULL,
  lambda1 = object$lambda$lambda1,
  lambda2 = object$lambda$lambda2,
  weights = NULL,
  ridge = object$lambda$ridge,
  verbose = FALSE,
  ...
)

```

**Arguments**

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>object</code>  | A "ssrhe.hessian.fit" object.                                                                                                                                                            |
| <code>y</code>       | New numeric response vector or matrix with one row per vertex. If NULL, the original response is reused.                                                                                 |
| <code>lambda1</code> | Nonnegative Hessian-energy penalty multiplier for $f^\top B f = \ A f\ _2^2$ .                                                                                                           |
| <code>lambda2</code> | Nonnegative supplemental-stabilizer multiplier for $f^\top B_S f$ . If positive, <code>stabilizer</code> must be TRUE. Currently supported only for <code>derivative.order = 2L</code> . |
| <code>weights</code> | Optional nonnegative observation weights. May be NULL, a vector of length <code>nrow(X)</code> , or a matrix with the same dimensions as <code>y</code> .                                |
| <code>ridge</code>   | Nonnegative diagonal ridge added to the linear system for numerical stabilization.                                                                                                       |
| <code>verbose</code> | Logical. If TRUE, print a short native construction message.                                                                                                                             |
| <code>...</code>     | Additional arguments are not currently supported.                                                                                                                                        |

**Value**

A list of class "ssrhe.hessian.refit".

**See Also**

[refit](#) for supported fitted objects and migration.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
fit <- fit.ssrhe.hessian.regression(
  X, X[, 1]^2, k = 6L, tangent.dim = 1L, lambda1 = 0.1
)
refit(fit, y = X[, 1]^3)
```

---

```
refit.ssrhe.hessian.l1.fit
```

*Refit SSRHE-Style Hessian L1 Regression*

---

**Description**

Reuses the  $A$  operator from [fit.ssrhe.hessian.l1.regression](#) to fit a new response or a new lambda grid without rebuilding local neighborhoods or local PCA Hessian rows.

**Usage**

```
## S3 method for class 'ssrhe.hessian.l1.fit'
refit(
  object,
  y = NULL,
  lambda.grid = object$lambda,
  lambda.selection = c("fixed", "cv"),
  weights = NULL,
  n.lambda = 40L,
  nfolds = 5L,
  fold.id = NULL,
  loss = c("mse", "mae"),
  selection = c("min", "one.se"),
  solver = c("genlasso", "admm", "auto"),
  row.scaling = c("none", "l2"),
  admm.rho = NULL,
  admm.maxiter = 2000L,
  admm.abstol = 1e-06,
  admm.reltol = 1e-05,
  maxsteps = 2000L,
  minlam = 0,
  approx = FALSE,
```

```

    rtol = 1e-07,
    btol = 1e-07,
    eps = 1e-04,
    verbose = FALSE,
    admm.adaptive.rho = TRUE,
    ...
)

```

## Arguments

|                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>object</code>                                                                                                                                 | A "ssrhe.hessian.l1.fit" object.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>y</code>                                                                                                                                      | Optional new numeric response vector. If NULL, the original response is reused.                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>lambda.grid</code>                                                                                                                            | Optional nonnegative lambda grid. For <code>lambda.selection = "fixed"</code> , this must contain exactly one value. For <code>lambda.selection = "cv"</code> , NULL builds a default grid from the fitted full-data generalized-lasso path.                                                                                                                                                                                                                     |
| <code>lambda.selection</code>                                                                                                                       | "cv" for observed-label K-fold cross-validation or "fixed" for a supplied fixed lambda.                                                                                                                                                                                                                                                                                                                                                                          |
| <code>weights</code>                                                                                                                                | Optional nonnegative observation weights. Missing responses automatically receive zero weight.                                                                                                                                                                                                                                                                                                                                                                   |
| <code>n.lambda</code>                                                                                                                               | Number of default lambda candidates when <code>lambda.grid = NULL</code> and <code>lambda.selection = "cv"</code> .                                                                                                                                                                                                                                                                                                                                              |
| <code>nfolds</code>                                                                                                                                 | Number of validation folds over observed positive-weight labels. Ignored when <code>fold.id</code> is supplied.                                                                                                                                                                                                                                                                                                                                                  |
| <code>fold.id</code>                                                                                                                                | Optional integer fold assignments of length <code>nrow(X)</code> .                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>loss</code>                                                                                                                                   | Validation loss, currently "mse" or "mae".                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>selection</code>                                                                                                                              | Selection rule. "min" chooses the smallest mean validation loss. "one.se" chooses the largest lambda within one standard error of the minimum.                                                                                                                                                                                                                                                                                                                   |
| <code>solver</code>                                                                                                                                 | Solver backend. "genlasso" uses the generalized-lasso path to propose a grid when needed, then refines every candidate with ADMM on the original objective. "admm" skips the path and requires an explicit grid. "auto" also proceeds when path construction fails, provided an explicit grid is supplied.                                                                                                                                                       |
| <code>row.scaling</code>                                                                                                                            | Optional row scaling for the penalty matrix before solving. "l2" scales nonzero rows of $A$ to unit Euclidean norm.                                                                                                                                                                                                                                                                                                                                              |
| <code>admm.rho</code> , <code>admm.maxiter</code> , <code>admm.abstol</code> , <code>admm.reltol</code>                                             | ADMM controls used by all backends, including refinement of generalized-lasso candidates. <code>admm.rho = NULL</code> chooses an initial algorithm penalty of $1 / \max(1, \text{mean}(\text{rowSums}(D^2)))$ , where $D$ is the penalty operator after any requested row scaling. A positive numeric value sets the initial penalty explicitly. This is separate from statistical regularization <code>lambda.grid</code> . The iteration limit is a hard cap. |
| <code>maxsteps</code> , <code>minlam</code> , <code>approx</code> , <code>rtol</code> , <code>btol</code> , <code>eps</code> , <code>verbose</code> | Controls passed to <b>genlasso</b> .                                                                                                                                                                                                                                                                                                                                                                                                                             |

```
admm.adaptive.rho
```

Logical; adapt the algorithm penalty during the first 1000 iterations by balancing tolerance-normalized primal and dual residuals. Set FALSE to hold the initial penalty fixed.

```
...
```

Additional arguments are not currently supported.

**Value**

A list of class "ssrhe.hessian.l1.refit".

**See Also**

[refit](#) for supported fitted objects and migration.

**Examples**

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
fit <- fit.ssrhe.hessian.l1.regression(
  X, X[, 1]^2, k = 6L, tangent.dim = 1L,
  lambda.grid = 0.05, lambda.selection = "fixed", solver = "admm"
)
refit(
  fit, y = X[, 1]^3, solver = "admm"
)
```

---

```
select.masked.graph.field
```

*Evaluate Nested Selection Among Candidate Graphs*

---

**Description**

Chooses the graph using inner-fold loss only, then assembles its outer-fold predictions. Only graphs with full inner-fold prediction coverage compete; unsupported outer predictions remain NA. Candidate vertex identities and ordering must agree. Final graph selection uses full-data inner CV.

**Usage**

```
select.masked.graph.field(comparison, study)
```

**Arguments**

|            |                                                            |
|------------|------------------------------------------------------------|
| comparison | Output from compare.masked.graph.fields.                   |
| study      | Name of the study used to choose graphs (for example HMP). |

**Value**

Selected graph per outer fold, outer predictions and coverage, participant-balanced MSE, final graph name, and its final fitted field.

slpl.tf.operator

*Construct A Synchronized LPL-TF Fixed Operator***Description**

Builds the fixed S-LPL-TF operator pair. The LPL component  $A_{\text{LPL}}$  is the accepted self-excluded LPL-TF residual operator from [lpl.tf.operator](#). The synchronization component  $C_{\text{sync}}$  compares inclusive local polynomial prediction maps over overlapping supports.

**Usage**

```
slpl.tf.operator(
  X,
  adj.list = NULL,
  weight.list = NULL,
  graph = NULL,
  graph.stage = "final",
  anchor.index = NULL,
  anchor.coordinates = NULL,
  degree = 2L,
  support.type = c("adaptive.radius", "knn", "fixed.radius"),
  support.size = NULL,
  radius = NULL,
  min.support = NULL,
  support.buffer = 3L,
  kernel = c("epanechnikov", "triangular", "gaussian", "tricube"),
  coordinate.method = c("coordinates", "local.pca"),
  chart.dim = NULL,
  support.metric = c("auto", "coordinates", "graph.geodesic"),
  auto.chart.support.metric = c("coordinates", "operator", "both"),
  auto.chart.selection.metric = c("coordinates", "operator"),
  row.normalize = c("l2", "none", "l1"),
  local.solver = c("auto", "normal.equations", "qr", "svd"),
  normal.equations.max.condition = 1e+08,
  duplicate.action = c("keep", "error"),
  drop.rank.deficient = TRUE,
  sync.row.normalize = c("l2"),
  sync.min.norm = 1e-12,
  verbose = FALSE,
  ...
)
```

**Arguments**

**X** Numeric coordinate matrix with one row per observation.

|                                          |                                                                                                                                                                                                                                                                                                                             |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list, weight.list</code>       | Optional supplied undirected graph adjacency and positive edge-length lists using 1-based vertex indices.                                                                                                                                                                                                                   |
| <code>graph</code>                       | Optional graph object returned by <code>dgraphs::create.rknn.graph()</code> or a list containing <code>adj.list/weight.list</code> or <code>adj_list/weight_list</code> .                                                                                                                                                   |
| <code>graph.stage</code>                 | Graph stage to extract from graph: "final", "raw", or "pruned".                                                                                                                                                                                                                                                             |
| <code>anchor.index</code>                | Optional integer vector of observed anchor/target indices. The current implementation supports observed anchors only.                                                                                                                                                                                                       |
| <code>anchor.coordinates</code>          | Reserved for later off-observation anchors; this implementation requires NULL.                                                                                                                                                                                                                                              |
| <code>degree</code>                      | Polynomial degree. The current implementation supports 0L, 1L, and 2L.                                                                                                                                                                                                                                                      |
| <code>support.type</code>                | Coordinate support rule. "knn" uses the nearest <code>support.size</code> candidates including the target before self exclusion. "adaptive.radius" uses the nearest <code>min.support</code> positive predictors after self exclusion. "fixed.radius" uses all points within <code>radius</code> .                          |
| <code>support.size</code>                | Positive integer for <code>support.type = "knn"</code> .                                                                                                                                                                                                                                                                    |
| <code>radius</code>                      | Positive coordinate radius for <code>support.type = "fixed.radius"</code> .                                                                                                                                                                                                                                                 |
| <code>min.support</code>                 | Minimum number of positive-weight predictors after self exclusion. If NULL, uses <code>choose(ncol(X) + degree, degree) + support.buffer</code> .                                                                                                                                                                           |
| <code>support.buffer</code>              | Nonnegative integer added to the local polynomial design size when choosing default supports.                                                                                                                                                                                                                               |
| <code>kernel</code>                      | Kernel used to weight local predictors by distance.                                                                                                                                                                                                                                                                         |
| <code>coordinate.method</code>           | Coordinate chart used for the local polynomial design. "coordinates" uses centered ambient coordinates; "local.pca" uses a deterministic local PCA chart on the selected support.                                                                                                                                           |
| <code>chart.dim</code>                   | Chart dimension. For "coordinates", this must be NULL or <code>ncol(X)</code> . For "local.pca", NULL defaults to <code>ncol(X)</code> . The special value "auto" estimates a single chart dimension from observed-coordinate local PCA spectra only, without using responses, truth values, latent coordinates, or labels. |
| <code>support.metric</code>              | Support-distance rule. "coordinates" uses Euclidean coordinate distances; "graph.geodesic" uses shortest-path distances in the supplied graph. "auto" uses graph geodesics when a graph is supplied and coordinate distances otherwise.                                                                                     |
| <code>auto.chart.support.metric</code>   | Support system used when <code>chart.dim = "auto"</code> . "coordinates" uses Euclidean coordinate neighborhoods, "operator" uses the resolved operator support metric, and "both" computes both diagnostics side by side.                                                                                                  |
| <code>auto.chart.selection.metric</code> | Which auto chart-dimension diagnostic to use for the fitted operator when <code>auto.chart.support.metric = "both"</code> . The default "coordinates" preserves historical behavior.                                                                                                                                        |
| <code>row.normalize</code>               | Complete-row normalization rule applied after setting target coefficient +1 and predictor coefficients <code>-h<sub>ij</sub></code> .                                                                                                                                                                                       |
| <code>local.solver</code>                | Local linear algebra rule.                                                                                                                                                                                                                                                                                                  |

```

normal.equations.max.condition
    Maximum condition number for local.solver = "auto" to use normal equations.
duplicate.action
    Duplicate coordinate policy.
drop.rank.deficient
    Logical. The current implementation requires TRUE.
sync.row.normalize
    Synchronization-row normalization. The current implementation uses "l2" by default.
sync.min.norm
    Minimum raw synchronization-row norm. Rows below this threshold are dropped with metadata.
verbose
    Logical. Reserved for diagnostic messages.
...
    Reserved.

```

**Value**

A list of class "slpl\_tf\_operator" containing A\_LPL, C\_sync, row metadata, diagnostics, settings, and the embedded "lpl\_tf\_operator".

**Examples**

```

X <- matrix(seq(0, 1, length.out = 18), ncol = 1)
slpl.tf.operator(X, degree = 1L, support.type = "knn",
  support.size = 7L, kernel = "gaussian")

```

---

|                 |                                              |
|-----------------|----------------------------------------------|
| smoother.matrix | <i>Extract A Conditional Smoother Matrix</i> |
|-----------------|----------------------------------------------|

---

**Description**

Extract the linear map from responses to predictions for a supported fitted model. R selects the method from the class of object.

**Usage**

```

smoother.matrix(object, ...)

## Default S3 method:
smoother.matrix(object, ...)

```

**Arguments**

```

object      An "lps" or "malps" fitted object.
...         Method-specific controls: check.tol for LPS; max.n and allow.robust for MALPS. Unrecognized arguments are errors.

```

## Details

`smoother.matrix.lps` requires a fixed-configuration Gaussian LPS fit using the R backend, the orthogonal polynomial drop basis, singleton tuning grids, and fixed chart dimension. It returns one row per evaluation point and checks that the matrix reproduces the stored raw fitted values.

`smoother.matrix.malps` returns a training-point matrix conditional on stored supports, charts, and weights. Its default size limit is `max.n = 1000L`. Robust fits are rejected unless `allow.robust = TRUE`; that option freezes the final robust weights and does not represent how those weights change with the response.

Neither method represents uncertainty from repeating model selection.

## Value

A numeric matrix mapping training responses to predictions. See the method help for dimensions and attributes.

## Migration

Replace `lps.smoother.matrix(fit, ...)` and `malps.smoother.matrix(fit, ...)` with `smoother.matrix(fit, ...)`. Existing method-specific arguments and restrictions are preserved. The former functions are no longer exported; their help aliases lead to the corresponding methods.

## See Also

`refit`, `lps.pointwise.band`, `malps.gcv`

## Examples

```
X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.malps(X, X[, 1]^2, degree = 1L,
               support.type = "knn", support.size = 8L)
S <- smoother.matrix(fit)
max(abs(as.vector(S %*% fit$y) - predict(fit)))
```

---

`smoother.matrix.lps`      *Extract the Linear-Smoother Matrix of a Fixed-Configuration LPS Fit*

---

## Description

For a fixed configuration (`singleton.support.grid`, `degree.grid`, and `kernel.grid`, with an explicit numeric chart dimension) the LPS fitted vector is linear in the response: `fitted = S %*% y` with `S` depending on `X`, the kernel weights, and the configuration, but not on `y`. This function reconstructs `S` analytically, one evaluation row at a time, by rebuilding each local fit's support, kernel weights, local chart, and design through the same internal routines `fit.lps()` used, and reading off the influence row of the local weighted least-squares solve.

**Usage**

```
## S3 method for class 'lps'
smoother.matrix(object, check.tol = 1e-10, ...)
```

**Arguments**

|           |                                                                                                                                                                                 |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object    | A fitted "lps" object from <code>fit.lps()</code> at a fixed configuration (see Details).                                                                                       |
| check.tol | Positive scalar: maximum allowed absolute discrepancy of the self-guard identity $S \% \% y == \text{fitted.values.raw}$ . Default $1e-10$ , the program's algebraic tolerance. |
| ...       | Additional arguments are not currently supported.                                                                                                                               |

**Details**

The extraction refuses configurations where the linearity premise fails or is unsupported: it requires `outcome.family = "gaussian"`, the R backend, `design.basis = "orthogonal.polynomial.drop"`, singleton grids (no data-driven selection: the CV-selected pipeline is *not* a linear smoother), and `coordinate.method = "coordinates" or "local.pca"` with an explicit numeric `chart.dim` (never "auto" or "local.auto").

Self-guard: before returning, the function verifies `max(abs(S %% y - fitted.values.raw)) <= check.tol` against the fit it was given (and that the NA patterns agree), so any divergence between the reconstruction and the estimator is a hard error, never a silent drift.

Local fits that fell back are represented honestly: a weighted-mean fallback (`unstable.action = "mean"`) contributes its exact linear row  $w / \text{sum}(w)$ ; an `unstable.action = "na"` non-fit contributes an all-NA row.

**Value**

A numeric matrix  $S$  with `nrow(object$X.eval)` rows and `nrow(object$X)` columns: row  $i$  holds the weights through which the training responses enter the prediction at evaluation point  $i$ .

**See Also**

`lps.pointwise.band()` for pointwise variances and confidence bands derived from  $S$ .

`smoother.matrix` for supported fitted objects and migration.

**Examples**

```
set.seed(1)
n <- 30
X <- matrix(runif(2 * n, -1, 1), ncol = 2)
y <- sin(pi * X[, 1]) + 0.1 * rnorm(n)
fit <- fit.lps(X, y, foldid = rep(1:2, length.out = n),
               support.grid = 12L, degree.grid = 1L,
               kernel.grid = "tricube", backend = "R",
               design.basis = "orthogonal.polynomial.drop",
               ridge.multiplier.grid = 0, ridge.condition.max = Inf,
               unstable.action = "na")
```

```

S <- smoother.matrix(fit)
max(abs(S %*% y - fit$fitted.values)) # ~1e-15: the linear identity
sum(diag(S))                         # effective degrees of freedom

```

---

smoother.matrix.malps *Construct The Conditional MALPS Smoother Matrix*

---

## Description

Constructs the exact training-point smoother matrix for a fitted [fit.malps](#) object, conditional on the stored supports, local charts, fitting weights, and averaging weights. For a non-robust MALPS fit or weighted refit with fixed supports, the returned matrix  $S$  satisfies  $\hat{y} = Sy$ . If the original fit used cross-validation, this matrix is conditional on the selected support profile; it does not account for the response-dependence of the model-selection step.

## Usage

```

## S3 method for class 'malps'
smoother.matrix(object, max.n = 1000L, allow.robust = FALSE, ...)

```

## Arguments

|              |                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------|
| object       | A "malps" object from <a href="#">fit.malps</a> or <a href="#">refit</a> .                                                   |
| max.n        | Maximum number of training observations for which a dense smoother matrix may be constructed. Use Inf to disable this guard. |
| allow.robust | Logical; allow fixed-final-weight linearization for fits with <code>robust.iterations &gt; 0L</code> .                       |
| ...          | Reserved for future extensions.                                                                                              |

## Details

Robust residual reweighting makes the effective fitting weights depend on the response. By default this function therefore rejects robust MALPS fits. Setting `allow.robust = TRUE` constructs the fixed-final-weight linearization, which reproduces the stored fit but should not be interpreted as the exact response-to-fit map.

## Value

A dense numeric  $n \times n$  matrix with attributes describing whether the matrix is conditional on support selection and whether it used a robust fixed-weight linearization.

## See Also

[smoother.matrix](#) for supported fitted objects and migration.

**Examples**

```

X <- matrix(seq(0, 1, length.out = 20), ncol = 1)
fit <- fit.malps(X, X[, 1]^2, degree = 1L,
               support.type = "knn", support.size = 8L)
S <- smoother.matrix(fit)
max(abs(S %*% fit$y - fit$fitted.values))

```

---

ssrhe.hessian.operator

*Construct an SSRHE-Style Local Hessian Energy Operator*


---

**Description**

Constructs the discrete local Hessian operator used by the semi-supervised regularization framework of Kim, Steinke, and Hein (2009). The constructor is intended as a package-facing, auditable R/C++ port of the reference behavior: it returns the sparse row operator  $A$ , the quadratic energy matrix  $B = A^T A$ , and optionally the supplemental stabilizer  $B_S$ .

**Usage**

```

ssrhe.hessian.operator(
  X,
  k = NULL,
  tangent.dim,
  nn.index = NULL,
  neighborhood.type = c("knn", "adaptive.radius", "supplied"),
  support.index = NULL,
  adaptive.k.scale = NULL,
  radius.rule = c("geomean", "max", "min"),
  radius.factor = 1.25,
  min.support = NULL,
  max.support = NULL,
  support.buffer = 2L,
  support.topup = c("nearest", "none"),
  tangent.dim.rule = c("fixed", "eigen.cumulative"),
  eigen.tolerance = 0.95,
  derivative.order = 2L,
  stabilizer = FALSE,
  pinv.tol = sqrt(.Machine$double.eps),
  local.solver = c("auto", "normal.equations", "svd", "qr"),
  normal.equations.max.condition = 10000,
  return.A = TRUE,
  return.B = TRUE,
  return.BS = stabilizer,
  return.sparse = TRUE,
  return.local.diagnostics = TRUE,
  return.timing = FALSE,

```

```

    verbose = FALSE
)

```

## Arguments

|                                         |                                                                                                                                                                                                                                                                                          |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>X</code>                          | Numeric matrix with one observation per row. Distances for the default neighborhood search are Euclidean distances in these coordinates.                                                                                                                                                 |
| <code>k</code>                          | Integer number of nearest neighbors per point, including the point itself, used when <code>neighborhood.type = "knn"</code> . This follows the SSRHE Matlab convention. For adaptive-radius neighborhoods, <code>k</code> may be omitted when <code>adaptive.k.scale</code> is supplied. |
| <code>tangent.dim</code>                | Integer tangent dimension used for the local PCA chart. Required when <code>tangent.dim.rule = "fixed"</code> . If <code>NULL</code> and <code>tangent.dim.rule = "eigen.cumulative"</code> , the dimension is selected locally from the PCA variance ratio.                             |
| <code>nn.index</code>                   | Optional integer matrix of neighbor indices, with <code>nrow(X)</code> rows and <code>k</code> columns. Each row must contain its center vertex. If <code>NULL</code> , Euclidean k-NN including self is computed in C++.                                                                |
| <code>neighborhood.type</code>          | Local support rule. "knn" uses the original rectangular self-including kNN neighborhoods. "adaptive.radius" builds variable-size supports from <code>dgraphs::create.rknn.graph</code> . "supplied" uses <code>support.index</code> directly.                                            |
| <code>support.index</code>              | Optional list of integer vectors, one per row of <code>X</code> . Each element gives a variable-size local support and must contain its center vertex.                                                                                                                                   |
| <code>adaptive.k.scale</code>           | Integer local-scale <code>k</code> used by <code>dgraphs::create.rknn.graph()</code> when <code>neighborhood.type = "adaptive.radius"</code> .                                                                                                                                           |
| <code>radius.rule, radius.factor</code> | Adaptive-radius graph parameters passed to <code>dgraphs::create.rknn.graph()</code> .                                                                                                                                                                                                   |
| <code>min.support</code>                | Optional minimum local support size for adaptive-radius supports. Defaults to the local quadratic design size plus <code>support.buffer</code> , clamped to <code>nrow(X)</code> .                                                                                                       |
| <code>max.support</code>                | Optional maximum local support size. Oversized adaptive supports are trimmed to the closest <code>max.support</code> vertices while keeping the center vertex.                                                                                                                           |
| <code>support.buffer</code>             | Nonnegative integer added to the local quadratic design size when choosing the default <code>min.support</code> .                                                                                                                                                                        |
| <code>support.topup</code>              | How undersized adaptive-radius supports are enlarged. "nearest" appends ambient nearest neighbors. "none" leaves supports unchanged and lets the C++ operator reject undersized supports.                                                                                                |
| <code>tangent.dim.rule</code>           | Either "fixed" or "eigen.cumulative".                                                                                                                                                                                                                                                    |
| <code>eigen.tolerance</code>            | Cumulative local PCA variance threshold used when <code>tangent.dim.rule = "eigen.cumulative"</code> and <code>tangent.dim = NULL</code> .                                                                                                                                               |
| <code>derivative.order</code>           | Integer derivative order of the local SSRHE-style operator. 2L is the original local Hessian energy. 3L is an experimental third-derivative energy whose rows estimate unique symmetric third-derivative tensor components with factorial and tensor-multiplicity scaling.               |

|                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| stabilizer                     | Logical. If TRUE, also construct the supplemental stabilizer matrix described in the SSRHE supplement. Currently supported only for <code>derivative.order = 2L</code> .                                                                                                                                                                                                                                                                                                                                                                                    |
| pinv.tol                       | Nonnegative tolerance multiplier for local pseudoinverses.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| local.solver                   | Local least-squares backend used to map local function values to derivative coefficients. "auto" is the default: it uses normal equations when the local design is full rank and has condition number no larger than <code>normal.equations.max.condition</code> , and otherwise falls back to SVD. "normal.equations" requests the normal-equation solve for full-rank local designs and falls back only on hard numerical failures. "svd" is the most stable reference path. "qr" uses pivoted QR and falls back to SVD for rank-deficient local designs. |
| normal.equations.max.condition | Positive condition-number guard used by <code>local.solver = "auto"</code> before accepting the normal-equation backend. The default is deliberately conservative because normal-equation solves square the effective condition number and order-3 near-minimum local supports can be numerically fragile.                                                                                                                                                                                                                                                  |
| return.A                       | Logical. If TRUE, return $A$ as a sparse matrix.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| return.B                       | Logical. If TRUE, return $B = A^\top A$ .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| return.BS                      | Logical. If TRUE and <code>stabilizer = TRUE</code> , return the supplemental stabilizer $B_S$ .                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| return.sparse                  | Logical. If TRUE, attach <b>Matrix</b> sparse matrices in addition to raw triplets.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| return.local.diagnostics       | Logical. If TRUE, compute additional R-side local chart diagnostics such as chart distortion and boundary asymmetry. These diagnostics are useful for operator audits, but can be skipped in fitting and cross-validation paths for speed.                                                                                                                                                                                                                                                                                                                  |
| return.timing                  | Logical. If TRUE, attach a phase-level elapsed time table to the returned operator. The timings separate R-side validation, neighborhood/support construction, native local operator construction, optional local diagnostics, sparse matrix assembly, and output finalization. For adaptive-radius neighborhoods, subphase timings are also attached in <code>neighborhoods\$timing</code> .                                                                                                                                                               |
| verbose                        | Logical. If TRUE, print a short native construction message.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## Details

For each point  $x_i$ , the operator constructs a local PCA chart from a self-including local support, projects the support into  $\mathbb{R}^m$ , and centers the local coordinates at  $x_i$ . A local quadratic model is then fitted with the intercept fixed at  $f_i$ :

$$f(x_j) - f(x_i) \approx \sum_{a \leq b} h_{ab} z_{ja} z_{jb} + \sum_a g_a z_{ja}.$$

The local linear map from function values to the quadratic coefficients is the Matlab-compatible

$$\text{RegMat} = X^+ - X^+ \text{IndMat},$$

where  $X$  is the reduced quadratic-plus-linear design and  $\text{IndMat}$  copies the center value into all local rows. Diagonal Hessian components are scaled by  $\sqrt{2}$  and off-diagonal components by 1, so that the row energy matches the reference Hessian energy. With `derivative.order = 3L`, the

reduced local design uses cubic monomials first, followed by quadratic and linear monomials. The returned rows estimate unique symmetric third-derivative components  $\partial_{abc}f$  with scale

$$s_{abc} = m_{abc}\sqrt{\mu_{abc}},$$

where  $m_{abc}$  is the factorial monomial-to-derivative multiplier (6 for  $aaa$ , 2 for  $aab$ , and 1 for three distinct indices) and  $\mu_{abc}$  is the ordered-tensor multiplicity (1, 3, or 6). Thus  $\|Af\|_2^2$  approximates the squared Frobenius norm of the full symmetric third-derivative tensor.

The returned  $A$  is the stacked sparse matrix of these local derivative coefficient rows. The  $\ell_2$  SSRHE penalty is

$$f^\top Bf = \|Af\|_2^2, \quad B = A^\top A.$$

Exposing  $A$  is useful for auditability and for future  $\ell_1$ -style variants based on  $\|Af\|_1$ .

Adaptive-radius graph construction uses **dgraphs**; fixed-k and supplied neighborhoods are package-local geosmooth paths.

## Value

A list of class "ssrhe.hessian.operator" containing:

- `A`: sparse local Hessian row operator, if requested;
- `B`: sparse quadratic energy matrix  $A^\top A$ , if requested;
- `BS`: optional supplemental stabilizer matrix;
- `A.triplet`, `BS.triplet`: raw sparse triplets;
- `nn.index`: neighbor index matrix for rectangular kNN neighborhoods, if used;
- `support.index`: list of actual local supports used by the backend;
- `neighborhoods`: support-construction diagnostics;
- `row.table`: one row per Hessian component row of  $A$ ;
- `diagnostics`: local design rank, condition, and PCA summaries;
- `local.diagnostics`: if requested, one row per center vertex with support size, design rank/condition, PCA variance, chart-distortion, boundary-asymmetry, and curvature-bias proxy diagnostics;
- `timing`: if requested, a phase-level elapsed-time table;
- `native.timing`: if requested, a C++ backend subphase timing table for native operator construction;
- `parity`: notes documenting reference-behavior conventions.

## References

Kim, K. I., Steinke, F., and Hein, M. (2009). Semi-supervised regression using Hessian energy with an application to semi-supervised dimensionality reduction. *Advances in Neural Information Processing Systems* 22. [https://papers.nips.cc/paper\\_files/paper/2009/hash/f4552671f8909587cf485ea990207f3b.html](https://papers.nips.cc/paper_files/paper/2009/hash/f4552671f8909587cf485ea990207f3b.html)

## Examples

```
X <- matrix(seq(0, 1, length.out = 12), ncol = 1)
ssrhe.hessian.operator(X, k = 6L, tangent.dim = 1L)
```

---

ssrhe.support.grid      *Build Candidate Support Profiles for SSRHE Adaptive-Radius Tuning*


---

### Description

Constructs a compact grid of adaptive-radius support profiles for use with `support.selection = "cv"` in SSRHE fitting functions. Each row gives an `adaptive.k.scale` value and a requested `min.support`. The grid is intentionally small by default because support selection nests operator construction inside response cross-validation.

### Usage

```
ssrhe.support.grid(
  n,
  tangent.dim,
  derivative.order = 2L,
  support.buffer = 2L,
  max.candidates = 8L
)
```

### Arguments

|                               |                                                                                                      |
|-------------------------------|------------------------------------------------------------------------------------------------------|
| <code>n</code>                | Number of observations.                                                                              |
| <code>tangent.dim</code>      | Tangent dimension used by the SSRHE local polynomial design.                                         |
| <code>derivative.order</code> | SSRHE derivative order, currently 2L or 3L.                                                          |
| <code>support.buffer</code>   | Nonnegative integer added to the local design size when constructing the smallest candidate support. |
| <code>max.candidates</code>   | Maximum number of candidate support profiles to return.                                              |

### Value

A data frame with columns `adaptive.k.scale`, `min.support`, and `max.support`. `max.support` is NA by default, meaning no truncation.

### Examples

```
ssrhe.support.grid(n = 50L, tangent.dim = 2L, max.candidates = 4L)
```

---

summary.harmonic\_smoother

*Summary Method for Harmonic Smoother Results*


---

## Description

Provides detailed summary statistics for harmonic smoother results, including the evolution of extrema counts and basin structure differences.

## Usage

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

## Arguments

|        |                                                    |
|--------|----------------------------------------------------|
| object | An object of class "harmonic_smoother".            |
| ...    | Further arguments passed to or from other methods. |

## Value

An object of class "summary.harmonic\_smoother" containing:

|                                      |                                                                                                 |
|--------------------------------------|-------------------------------------------------------------------------------------------------|
| stable_iteration, stability_detected | Extrema-stability detection, separate from numerical convergence; iteration is NA if undetected |
| converged, status, num_iterations    | Numerical relaxation status                                                                     |
| iterations_recorded                  | Total number of recorded iterations                                                             |
| initial_extrema                      | Number of extrema at the first iteration                                                        |
| initial_maxima                       | Number of maxima at the first iteration                                                         |
| initial_minima                       | Number of minima at the first iteration                                                         |
| final_extrema                        | Number of extrema at the final iteration                                                        |
| final_maxima                         | Number of maxima at the final iteration                                                         |
| final_minima                         | Number of minima at the final iteration                                                         |
| extrema_reduction                    | Reduction in number of extrema                                                                  |
| topology_diff_summary                | Summary statistics of topology differences                                                      |

## See Also

[harmonic.smoother](#), [print.summary.harmonic\\_smoother](#)

**Examples**

```
fit <- structure(
  list(
    stable_iteration = 2L,
    i_basins = list(matrix(c(1, 1), ncol = 2)),
    topology_differences = numeric()
  ),
  class = "harmonic_smoother"
)
summary(fit)
```

---

support.masked.graph.field

*Summarize Measurement Support for a Graph Field*

---

**Description**

Reports graph-hop distance to the nearest observed vertex and the number of distinct measured participants at each vertex and its immediate neighbors. These are descriptive support measures, not uncertainty intervals or a guarantee of transportability. Vertex order follows the supplied operator.

**Usage**

```
support.masked.graph.field(operator, vertex, group)
```

**Arguments**

|          |                                                          |
|----------|----------------------------------------------------------|
| operator | An unnormalized metric.graph.lowpass.operator.           |
| vertex   | Integer vertex indices of observed responses.            |
| group    | Nonmissing participant identifiers for the observations. |

**Value**

A data frame with nearest.observation.hops (Inf if unreachable), local.participants and component.supported for each vertex.

---

`synthetic.dataset.checksum`*Compute the canonical content checksum of a synthetic dataset*

---

**Description**

Compute the canonical content checksum of a synthetic dataset

**Usage**

```
synthetic.dataset.checksum(x)
```

**Arguments**

`x`                    A `synthetic_dataset`.

**Value**

A lowercase SHA-256 string.

**Examples**

```
x <- materialize.synthetic(synthetic.registry.spec("G1"), n = 20L, seed = 1L)
synthetic.dataset.checksum(x)
```

---

`synthetic.registry.ids`*List maintained synthetic recipe IDs*

---

**Description**

List maintained synthetic recipe IDs

**Usage**

```
synthetic.registry.ids()
```

**Value**

Character recipe IDs.

**Examples**

```
head(synthetic.registry.ids())
```

synthetic.registry.seed

*Resolve the legacy seed for an SSRHE registry recipe*

---

### Description

Resolve the legacy seed for an SSRHE registry recipe

### Usage

```
synthetic.registry.seed(  
    recipe.id,  
    replicate = 1L,  
    n = NULL,  
    base.seed = 273001L  
)
```

### Arguments

|           |                                                       |
|-----------|-------------------------------------------------------|
| recipe.id | A one-dimensional, flat, or quadform SSRHE recipe ID. |
| replicate | Positive replicate number.                            |
| n         | Sample size. Required for flat and quadform recipes.  |
| base.seed | One-dimensional suite base seed.                      |

### Value

A validated integer seed.

### Examples

```
synthetic.registry.seed("S16.V1", replicate = 2L)
```

---

synthetic.registry.spec

*Resolve a maintained synthetic recipe*

---

### Description

Default recipes are reconstructed from normalized component foreign keys. Parameter overrides are accepted only for the legacy G-family compatibility surface and remain content-bound, non-frozen specifications.

### Usage

```
synthetic.registry.spec(recipe.id, parameters = list())
```

**Arguments**

|            |                                     |
|------------|-------------------------------------|
| recipe.id  | One of synthetic.registry.ids().    |
| parameters | Optional named parameter overrides. |

**Value**

A validated synthetic\_spec.

**Examples**

```
synthetic.registry.spec("G1")
```

---

|                                         |
|-----------------------------------------|
| synthetic.response.bernoulli            |
| <i>Bernoulli response specification</i> |

---

**Description**

Bernoulli response specification

**Usage**

```
synthetic.response.bernoulli(minimum.positive = 0L, maximum.attempts = 1L)
```

**Arguments**

|                  |                                      |
|------------------|--------------------------------------|
| minimum.positive | Minimum accepted positive responses. |
| maximum.attempts | Maximum complete redraw attempts.    |

**Value**

A synthetic response component.

**Examples**

```
synthetic.response.bernoulli(minimum.positive = 1L, maximum.attempts = 5L)
```

synthetic.response.clustered.gaussian  
*Clustered Gaussian response specification*

---

**Description**

Clustered Gaussian response specification

**Usage**

```
synthetic.response.clustered.gaussian(residual.sd, intraclass.correlation)
```

**Arguments**

residual.sd      Residual standard deviation.  
intraclass.correlation  
                  Intraclass correlation in  $[0, 1)$ .

**Value**

A synthetic response component.

**Examples**

```
synthetic.response.clustered.gaussian(0.2, 0.3)
```

---

synthetic.response.gaussian  
*Gaussian response specification*

---

**Description**

Gaussian response specification

**Usage**

```
synthetic.response.gaussian(sd)
```

**Arguments**

sd                      Response standard deviation.

**Value**

A synthetic response component.

**Examples**

```
synthetic.response.gaussian(sd = 0.2)
```

---

```
synthetic.response.heteroskedastic.gaussian
```

*Heteroskedastic Gaussian response specification*

---

**Description**

Heteroskedastic Gaussian response specification

**Usage**

```
synthetic.response.heteroskedastic.gaussian(base.sd, truth.multiplier)
```

**Arguments**

`base.sd`            Base standard deviation.  
`truth.multiplier`    Multiplier applied to truth.

**Value**

A synthetic response component.

**Examples**

```
synthetic.response.heteroskedastic.gaussian(0.1, 0.2)
```

---

```
synthetic.response.laplace.outlier
```

*Laplace response with Gaussian contamination*

---

**Description**

Laplace response with Gaussian contamination

**Usage**

```
synthetic.response.laplace.outlier(  
  laplace.scale,  
  outlier.fraction,  
  outlier.sd,  
  minimum.outliers = 0L,  
  count.rounding = "round",  
  laplace.algorithm = "uniform.inverse.v1"  
)
```

**Arguments**

`laplace.scale`    Laplace scale.  
`outlier.fraction`    Fraction contaminated.  
`outlier.sd`    Contamination standard deviation.  
`minimum.outliers`    Minimum contamination count.  
`count.rounding`    Count policy.  
`laplace.algorithm`    Versioned Laplace algorithm.

**Value**

A synthetic response component.

**Examples**

```
synthetic.response.laplace.outlier(0.1, 0.05, 1)
```

---

```
synthetic.sampling.stratified
```

*Legacy G4 stratified sampling specification*

---

**Description**

Legacy G4 stratified sampling specification

**Usage**

```
synthetic.sampling.stratified(
  fraction.a = 0.5,
  algorithm = c("stratum.sequential.v1", "legacy.g4.stratum.sequential.v1")
)
```

**Arguments**

`fraction.a`    Fraction allocated to stratum A.  
`algorithm`    Versioned traversal algorithm.

**Value**

A synthetic sampling component.

**Examples**

```
synthetic.sampling.stratified(fraction.a = 0.6)
```

---

|                |                                                |
|----------------|------------------------------------------------|
| synthetic.spec | <i>Assemble a synthetic-data specification</i> |
|----------------|------------------------------------------------|

---

## Description

Combines draw-free geometry, sampling, truth, and response components into one canonically hashable specification.

## Usage

```
synthetic.spec(  
  geometry,  
  sampling,  
  truth,  
  response,  
  recipe.id = NULL,  
  registry.tag = NULL,  
  compatibility = NULL,  
  metadata = list()  
)
```

## Arguments

|               |                                                      |
|---------------|------------------------------------------------------|
| geometry      | A synthetic geometry component.                      |
| sampling      | A synthetic sampling component.                      |
| truth         | A synthetic truth component.                         |
| response      | A synthetic response component.                      |
| recipe.id     | Optional readable recipe label.                      |
| registry.tag  | Optional stable experimental-plan tag.               |
| compatibility | Optional serializable legacy compatibility metadata. |
| metadata      | Optional serializable scientific metadata.           |

## Value

A synthetic\_spec.

## Examples

```
spec <- synthetic.spec(  
  dgraphs::synthetic.circle(), dgraphs::synthetic.sampling.uniform.interval(0, 2 * pi),  
  synthetic.truth.named("helix.sin.v1"), synthetic.response.gaussian(0.1)  
)  
spec$specification.sha256
```

---

```
synthetic.truth.gaussian.mixture
```

*Specify a Gaussian-mixture truth*

---

## Description

Specify a Gaussian-mixture truth

## Usage

```
synthetic.truth.gaussian.mixture(  
  centers,  
  scales,  
  weights,  
  normalize = c("none", "sample.max"),  
  evaluation.coordinates = "latent"  
)
```

## Arguments

|                                     |                                                                |
|-------------------------------------|----------------------------------------------------------------|
| <code>centers</code>                | Component centers, one per row.                                |
| <code>scales</code>                 | Positive isotropic standard deviations or covariance matrices. |
| <code>weights</code>                | Nonnegative mixture weights.                                   |
| <code>normalize</code>              | Normalization scope.                                           |
| <code>evaluation.coordinates</code> | Coordinates on which to evaluate.                              |

## Value

A synthetic truth component.

## Examples

```
synthetic.truth.gaussian.mixture(  
  centers = rbind(-1, 1), scales = c(0.5, 0.5), weights = c(1, 1)  
)
```

---

synthetic.truth.logit *Specify a sinusoidal finite-design logit truth*

---

**Description**

Specify a sinusoidal finite-design logit truth

**Usage**

```
synthetic.truth.logit(  
  amplitude = 1.5,  
  target.prevalence = 0.5,  
  clip = c(0.05, 0.95)  
)
```

**Arguments**

|                   |                                       |
|-------------------|---------------------------------------|
| amplitude         | Sinusoidal log-odds amplitude.        |
| target.prevalence | Target mean pre-clipping probability. |
| clip              | Probability bounds.                   |

**Value**

A synthetic truth component.

**Examples**

```
synthetic.truth.logit(amplitude = 1, target.prevalence = 0.4)
```

---

synthetic.truth.named *Specify a versioned named truth evaluator*

---

**Description**

Specify a versioned named truth evaluator

**Usage**

```
synthetic.truth.named(id, parameters = list())
```

**Arguments**

|            |                                    |
|------------|------------------------------------|
| id         | Evaluator ID.                      |
| parameters | Serializable evaluator parameters. |

**Value**

A synthetic truth component.

**Examples**

```
synthetic.truth.named("intrinsic.linear.first.v1")
```

---

```
synthetic.truth.occupation.mixture
```

*Specify a finite-design occupation-mixture truth*

---

**Description**

The analytic Gaussian-mixture density is transformed to Bernoulli occupation probabilities by  $\text{probability.maximum} * \text{density}^{\text{gamma}} / \max(\text{density}^{\text{gamma}})$ .

**Usage**

```
synthetic.truth.occupation.mixture(  
  centers,  
  covariances,  
  weights,  
  gamma = 1,  
  probability.maximum = 0.65,  
  normalization = "design.maximum"  
)
```

**Arguments**

|                     |                                          |
|---------------------|------------------------------------------|
| centers             | Component centers, one per row.          |
| covariances         | Positive-definite covariance matrices.   |
| weights             | Nonnegative mixture weights.             |
| gamma               | Positive density-shape exponent.         |
| probability.maximum | Maximum realized occupation probability. |
| normalization       | Currently "design.maximum".              |

**Value**

A finite-design synthetic truth component.

**Examples**

```
synthetic.truth.occupation.mixture(  
  centers = rbind(c(-1, 0), c(1, 0)),  
  covariances = list(diag(2), diag(2)), weights = c(1, 1)  
)
```

---

```
synthetic.truth.polynomial
```

*Specify a polynomial truth*

---

**Description**

Specify a polynomial truth

**Usage**

```
synthetic.truth.polynomial(coefficients, evaluation.coordinates = "latent")
```

**Arguments**

`coefficients`      Named polynomial coefficients.  
`evaluation.coordinates`  
                     Coordinates on which to evaluate.

**Value**

A synthetic truth component.

**Examples**

```
synthetic.truth.polynomial(c(b0 = 1, b1 = 2, b11 = -0.5))
```

---

```
transported.graph.hessian.operator
```

*Construct a Transported Graph Hessian Operator*

---

**Description**

Builds an experimental transported-Hessian graph operator without fitting a response. This diagnostic layer constructs directed edge differences, matches directions across a base dart, assembles a second- or third-difference diagnostic operator, and returns diagnostics that make the transport rule auditable.

**Usage**

```
transported.graph.hessian.operator(  
  adj.list,  
  weight.list = NULL,  
  transport.order = 2L,  
  transport.rule = c("exact.coordinate", "local.embedding.soft", "edge.angle.hard",  
    "edge.angle.soft", "regression.gradient"),
```

```

coordinates = NULL,
direction.labels = NULL,
polynomial.probes = NULL,
local.embedding.method = c("auto", "coordinates", "grip.edge.kk", "mds.edge.kk",
    "cmdscale"),
local.embedding.dim = 2L,
local.disk.hops = 1L,
local.max.vertices = 50L,
return.sparse = TRUE,
tol = 1e-08,
soft.angle.scale = NULL,
soft.length.scale = NULL,
soft.bandwidth = 0.25,
max.match.angle = NULL,
max.length.relative.error = NULL,
min.match.margin = NULL,
max.effective.matches = NULL,
match.threshold.rule = c("none", "fixed", "local.quantile", "local.robust.z"),
match.score.quantile = 0.25,
match.margin.quantile = 0.5,
min.best.score.z = 1,
min.margin.z = 0,
max.effective.match.fraction = NULL,
edge.angle.scale = NULL,
edge.length.scale = NULL,
edge.angle.bandwidth = 0.35,
edge.angle.max.angle.difference = NULL,
edge.angle.max.length.relative.error = NULL,
gradient.coordinate.method = c("coordinates", "local.embedding"),
gradient.embedding.method = c("grip.edge.kk", "mds.edge.kk", "cmdscale"),
gradient.embedding.dim = NULL,
gradient.disk.hops = local.disk.hops,
gradient.max.vertices = local.max.vertices,
gradient.disk.rule = c("hops", "metric.diameter.fraction", "metric.local.scale"),
gradient.disk.radius.fraction = 0.1,
gradient.disk.local.scale.method = c("knn.distance", "median.incident.length",
    "quantile.incident.length"),
gradient.disk.local.scale.k = 8L,
gradient.disk.local.scale.quantile = 0.75,
gradient.disk.local.scale.multiplier = 2,
gradient.disk.min.vertices = 0L,
gradient.chart.selection = c("fixed", "adaptive"),
gradient.embedding.candidates = c("cmdscale", "mds.edge.kk"),
gradient.disk.rule.candidates = c("hops", "metric.diameter.fraction",
    "metric.local.scale"),
gradient.disk.hops.candidates = 1:5,
gradient.disk.radius.fraction.candidates = c(0.05, 0.075, 0.1, 0.15, 0.2),
gradient.disk.local.scale.multiplier.candidates = c(1, 1.5, 2, 3, 4),

```

```

    gradient.ridge = 1e-08,
    gradient.quadratic.disk.hops = 2L,
    gradient.quadratic.max.vertices = gradient.max.vertices
)

```

## Arguments

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>adj.list</code>               | List of integer neighbor vectors using 1-based vertex indices. The graph must be undirected.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>weight.list</code>            | Optional list of positive edge lengths parallel to <code>adj.list</code> . If <code>NULL</code> , all edge lengths are one.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>transport.order</code>        | Integer scalar, either 2L or 3L. 2L constructs the transported Hessian-like second-difference operator. 3L currently constructs a supplied-coordinate regression-quadratic third-difference operator and is supported only for <code>transport.rule = "regression.gradient"</code> with <code>gradient.coordinate.method = "coordinates"</code> .                                                                                                                                                                                                                                                                                              |
| <code>transport.rule</code>         | Character scalar. "exact.coordinate" uses exact direction labels. "local.embedding.soft" uses coordinate direction vectors and softmax transport weights. "edge.angle.hard" and "edge.angle.soft" use coordinate-derived edge angles relative to the base dart. "regression.gradient" compares locally estimated gradient components.                                                                                                                                                                                                                                                                                                          |
| <code>coordinates</code>            | Optional numeric matrix with one row per vertex. For <code>transport.rule = "exact.coordinate"</code> , coordinates must define axis-aligned graph edges so direction labels can be inferred. Coordinates are required by "edge.angle.hard" and "edge.angle.soft".                                                                                                                                                                                                                                                                                                                                                                             |
| <code>direction.labels</code>       | Optional list parallel to <code>adj.list</code> ; each entry gives the exact direction label for the corresponding outgoing dart. When supplied, these labels override labels inferred from coordinates.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>polynomial.probes</code>      | Optional numeric vector or matrix with one row per vertex. If supplied, residual diagnostics $\ AP\ _F/\ P\ _F$ are reported for the transported Hessian matrix $A$ .                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>local.embedding.method</code> | Character scalar controlling the coordinates used by <code>transport.rule = "local.embedding.soft"</code> . "auto" uses supplied coordinates when available and otherwise tries "grip.edge.kk". "coordinates" uses supplied coordinates directly. "grip.edge.kk" builds a local graph disk and prefers weighted GRIP followed by true edge-KK optimization when the installed <b>grip</b> exposes <code>weighted.grip()</code> and <code>edge.kk()</code> . Older compatibility names are used only as fallbacks. "mds.edge.kk" uses classical MDS followed by the same edge-KK optimizer when available. "cmdscales" uses classical MDS only. |
| <code>local.embedding.dim</code>    | Positive integer embedding dimension for local graph-disk embeddings.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>local.disk.hops</code>        | Non-negative integer hop radius around each base dart used by graph-derived local embedding methods.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>local.max.vertices</code>     | Positive integer cap on local disk size. If a disk is larger, the nearest vertices by hop distance are retained.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

|                                             |                                                                                                                                                                                                                                                                                 |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| return.sparse                               | Logical. If TRUE, attach <b>Matrix</b> sparse matrices to the returned payloads.                                                                                                                                                                                                |
| tol                                         | Positive numeric tolerance for coordinate-axis direction inference.                                                                                                                                                                                                             |
| soft.angle.scale, soft.length.scale         | Optional positive numeric scales for the soft transport angular and length score components. If NULL, robust global defaults are estimated from candidate dart comparisons.                                                                                                     |
| soft.bandwidth                              | Positive numeric softmax bandwidth $\tau$ . Smaller values make the soft rule closer to hard nearest-direction matching.                                                                                                                                                        |
| max.match.angle                             | Optional non-negative numeric angle threshold in radians. For soft transport, candidate Hessian rows whose best match has angle larger than this threshold are dropped.                                                                                                         |
| max.length.relative.error                   | Optional non-negative numeric threshold for the best match's relative edge-length error, $ \ell_{\text{matched}} - \ell_{\text{direction}} /\ell_{\text{direction}}$ . Candidate rows above the threshold are dropped.                                                          |
| min.match.margin                            | Optional non-negative numeric threshold for the difference between the second-best and best soft-match scores. Candidate rows with smaller margins are dropped.                                                                                                                 |
| max.effective.matches                       | Optional positive numeric threshold for $\exp(H)$ , where $H$ is the softmax transport entropy. Candidate rows with more diffuse matches are dropped.                                                                                                                           |
| match.threshold.rule                        | Character scalar. "none" and "fixed" apply only the explicit fixed gates above. "local.quantile" also gates rows by local candidate-score and margin quantiles. "local.robust.z" also gates rows by robust local z-scores for best-match score and margin.                      |
| match.score.quantile, match.margin.quantile | Quantile thresholds used by match.threshold.rule = "local.quantile". The best score must lie at or below match.score.quantile; the best-vs-second margin must lie at or above match.margin.quantile.                                                                            |
| min.best.score.z, min.margin.z              | Non-negative robust-z thresholds used by match.threshold.rule = "local.robust.z". A larger best-score z-score means the best candidate is unusually low-scoring relative to local alternatives. A larger margin z-score means the best-vs-second separation is unusually large. |
| max.effective.match.fraction                | Optional positive numeric threshold for $\exp(H)/n_{\text{candidate}}$ . Candidate rows whose softmax mass is too diffuse across the local target directions are dropped.                                                                                                       |
| edge.angle.scale, edge.length.scale         | Optional positive numeric scales for edge-angle matching. If NULL, robust global medians are estimated from candidate angle differences and length differences.                                                                                                                 |
| edge.angle.bandwidth                        | Positive numeric softmax bandwidth used only by transport.rule = "edge.angle.soft".                                                                                                                                                                                             |
| edge.angle.max.angle.difference             | Optional non-negative threshold, in radians, for accepting an edge-angle match.                                                                                                                                                                                                 |

`edge.angle.max.length.relative.error`  
Optional non-negative threshold for accepting an edge-angle match based on relative edge-length error.

`gradient.coordinate.method`  
Character scalar used only by `transport.rule = "regression.gradient"`. "coordinates" uses the supplied global coordinates. "local.embedding" estimates the two endpoint gradients of each base dart in one shared graph-derived local chart.

`gradient.embedding.method`  
Character scalar used by `gradient.coordinate.method = "local.embedding"`. The choices match the graph-derived local embedding backends used by `local.embedding.soft`.

`gradient.embedding.dim`  
Optional positive integer dimension for graph-derived regression-gradient local charts. If NULL, the supplied coordinate dimension is used when available, otherwise `local.embedding.dim`.

`gradient.disk.hops`  
Non-negative integer hop radius for graph-derived regression-gradient local charts.

`gradient.max.vertices`  
Positive integer cap on graph-derived regression-gradient local chart size.

`gradient.disk.rule`  
Character scalar controlling graph-derived regression-gradient disk construction. "hops" uses `gradient.disk.hops`. "metric.diameter.fraction" uses a two-center graph-geodesic disk whose metric radius is `gradient.disk.radius.fraction` times the graph diameter. "metric.local.scale" uses a two-center graph-geodesic disk whose metric radius is `gradient.disk.local.scale.multiplier` times the larger endpoint local scale.

`gradient.disk.radius.fraction`  
Positive numeric scalar used by `gradient.disk.rule = "metric.diameter.fraction"`.

`gradient.disk.local.scale.method`  
Character scalar controlling the per-vertex metric scale used by `gradient.disk.rule = "metric.local.scale"`. "knn.distance" uses the weighted graph distance to the `gradient.disk.local.scale.k`-th nearest vertex. "median.incident.length" and "quantile.incident.length" use incident edge-length summaries.

`gradient.disk.local.scale.k`  
Positive integer neighborhood index used by `gradient.disk.local.scale.method = "knn.distance"`.

`gradient.disk.local.scale.quantile`  
Numeric probability in  $[0, 1]$  used by `gradient.disk.local.scale.method = "quantile.incident.length"`.

`gradient.disk.local.scale.multiplier`  
Positive numeric scalar multiplier for `gradient.disk.rule = "metric.local.scale"`.

`gradient.disk.min.vertices`  
Non-negative integer lower target for metric disk coverage. Metric disks whose radius selects too few vertices are expanded to include at least this many nearest vertices, subject to `gradient.max.vertices`.

`gradient.chart.selection`  
Character scalar used only by graph-derived regression-gradient transport. "fixed" uses `gradient.embedding.method` and `gradient.disk.hops`. "adaptive"

tries `gradient.embedding.candidates` crossed with `gradient.disk.hops.candidates` for each base dart and chooses the chart with the lowest graph-only diagnostic score.

`gradient.embedding.candidates`  
Character vector of graph-derived local embedding backends considered by `gradient.chart.selection`  
= "adaptive".

`gradient.disk.rule.candidates`  
Character vector of disk construction rules considered by `gradient.chart.selection`  
= "adaptive".

`gradient.disk.hops.candidates`  
Non-negative integer vector of hop radii considered by `gradient.chart.selection`  
= "adaptive".

`gradient.disk.radius.fraction.candidates`  
Positive numeric vector of graph-diameter fractions considered by `gradient.chart.selection`  
= "adaptive".

`gradient.disk.local.scale.multiplier.candidates`  
Positive numeric vector of local-scale multipliers considered by `gradient.chart.selection`  
= "adaptive".

`gradient.ridge` Non-negative ridge parameter used by `transport.rule = "regression.gradient"` when inverting local weighted least-squares normal equations.

`gradient.quadratic.disk.hops`  
Non-negative integer hop radius for the supplied-coordinate local quadratic regressions used by `transport.order = 3L`.

`gradient.quadratic.max.vertices`  
Positive integer cap on local quadratic-regression disk size for `transport.order = 3L`.

## Details

For a dart  $u \rightarrow v$ , the first difference is

$$\delta_{u \rightarrow v} f = \frac{f(v) - f(u)}{\ell_{uv}}.$$

For a base dart  $u \rightarrow u'$  and a matched outgoing direction  $u \rightarrow v \leftrightarrow u' \rightarrow v'$ , the transported Hessian row is

$$\delta_{u' \rightarrow v'} f - \delta_{u \rightarrow v}.$$

The reference rule, `transport.rule = "exact.coordinate"`, matches directions with identical coordinate labels. Labels may be supplied directly through `direction.labels`, or inferred from axis-aligned coordinates. The first general metric rule, `transport.rule = "local.embedding.soft"`, uses supplied coordinates to softly match directions by angle and edge length. The edge-angle rules, `"edge.angle.hard"` and `"edge.angle.soft"`, match directions by comparing their angle relative to the transported base edge. `transport.rule = "regression.gradient"` estimates local least-squares gradients and compares gradient components across graph darts. Those gradients can be computed either from supplied coordinates or from a graph-derived local embedding built in a shared disk around each base dart.

**Value**

A list of class "transported.graph.hessian.operator" with the canonical graph, directed-edge table, first-difference matrix, transported Hessian matrix, row metadata, dropped candidate rows, and diagnostics. For graph-derived regression-gradient transport with supplied coordinates, the embedding diagnostics also include synthetic chart-quality checks against those ambient coordinates.

**Examples**

```
adj <- list(2L, c(1L, 3L), c(2L, 4L), 3L)
weights <- list(1, c(1, 1), c(1, 1), 1)
x <- 0:3
transported.graph.hessian.operator(
  adj, weights, coordinates = matrix(x, ncol = 1),
  polynomial.probes = cbind(1, x, x^2)
)
```

---

```
validate.synthetic.dataset
```

*Validate a canonical synthetic dataset*

---

**Description**

Validate a canonical synthetic dataset

**Usage**

```
validate.synthetic.dataset(x)
```

**Arguments**

x                    A synthetic\_dataset.

**Value**

x, invisibly.

**Examples**

```
x <- materialize.synthetic(synthetic.registry.spec("G1"), n = 20L, seed = 1L)
validate.synthetic.dataset(x)
```

# Index

`apply.metric.graph.lowpass.path`, 5  
`as.data.frame.synthetic_dataset`, 7  
  
`bootstrap.malps`, 7  
  
`compare.masked.graph.fields`, 9  
`compare.synthetic.dataset`, 9  
`cv.masked.graph.field`, 10  
  
`fit.chart.kernel`, 11  
`fit.density`, 4, 13, 55, 56, 86  
`fit.graph.trend.filtering`, 4, 15  
`fit.local.likelihood`, 17  
`fit.lpl.tf`, 4, 20  
`fit.lps`, 4, 12, 19, 22, 37–39, 84  
`fit.lps()`, 66, 69, 116  
`fit.malps`, 4, 7, 8, 28, 71, 92, 104, 105, 117  
`fit.masked.graph.field`, 32  
`fit.metric.graph.lowpass`, 4, 33, 84  
`fit.ps.lps`, 35, 84  
`fit.pttf.trend.filtering`, 39  
`fit.slpl.tf`, 4, 42  
`fit.ssrhe.hessian.l1.regression`, 4, 41, 44, 109  
`fit.ssrhe.hessian.regression`, 4, 48, 49, 108  
`fit.subject.od`, 4, 54  
`fitted()`, 54  
`fitted.chart_kernel(model-values)`, 81  
`fitted.graph.trend.filtering.fit(model-values)`, 81  
`fitted.harmonic_smoother(model-values)`, 81  
`fitted.local_likelihood(model-values)`, 81  
`fitted.lpl_tf(model-values)`, 81  
`fitted.lps(model-values)`, 81  
`fitted.malps(model-values)`, 81  
`fitted.metric.graph.lowpass.fit(model-values)`, 81  
  
`fitted.metric.graph.lowpass.refit(model-values)`, 81  
`fitted.ps_lps(model-values)`, 81  
`fitted.pttf.trend.filtering.fit(model-values)`, 81  
`fitted.slpl_tf(model-values)`, 81  
`fitted.ssrhe.hessian.fit(model-values)`, 81  
`fitted.ssrhe.hessian.l1.fit(model-values)`, 81  
`fitted.ssrhe.hessian.refit(model-values)`, 81  
  
`geosmooth(geosmooth-package)`, 4  
`geosmooth-package`, 4  
`get.region.boundary`, 56, 87  
`graph.trend.filtering.operator`, 58  
`graphics::plot()`, 89  
  
`harmonic.smoother`, 57, 59, 87, 88, 95, 123  
  
`lpl.tf.operator`, 21, 22, 62, 112  
`lps.backend.diagnostics`, 64  
`lps.backend.diagnostics()`, 27  
`lps.grouped.foldid`, 65  
`lps.grouped.foldid()`, 67, 68  
`lps.nested.cv`, 66  
`lps.nested.cv()`, 27, 66  
`lps.pointwise.band`, 68, 115  
`lps.pointwise.band()`, 116  
`lps.smoother.matrix(smoother.matrix.lps)`, 115  
  
`malps.gcv`, 70, 115  
`malps.smoother.matrix(smoother.matrix.malps)`, 117  
`materialize.synthetic`, 4, 71  
`materialize.synthetic.instance`, 72  
`metric.graph.heat.eta.grid`, 73  
`metric.graph.heat.extend.lower`, 74

metric.graph.lowpass.basis, 75  
 metric.graph.lowpass.operator, 77  
 model-summary, 79  
 model-values, 81  
  
 normalize.density, 4, 11, 17, 55, 56, 84  
  
 perform.harmonic.smoothing, 57, 60, 61, 86  
 plot.harmonic\_smoother, 61, 88  
 plot.synthetic\_dataset, 89  
 predict.lpl\_tf, 90  
 predict.lps, 91  
 predict.malps, 91  
 predict.slpl\_tf, 92  
 print.chart\_kernel(model-summary), 79  
 print.density\_fit(model-summary), 79  
 print.graph.trend.filtering.fit, 93  
 print.harmonic\_smoother, 95  
 print.local\_likelihood(model-summary), 79  
 print.lpl\_tf(model-summary), 79  
 print.lpl\_tf\_operator  
     (print.graph.trend.filtering.fit), 93  
 print.lps  
     (print.graph.trend.filtering.fit), 93  
 print.malps  
     (print.graph.trend.filtering.fit), 93  
 print.malps\_bootstrap  
     (print.graph.trend.filtering.fit), 93  
 print.metric.graph.lowpass.fit  
     (print.graph.trend.filtering.fit), 93  
 print.metric.graph.lowpass.refit  
     (print.graph.trend.filtering.fit), 93  
 print.ps\_lps(model-summary), 79  
 print.pttf.trend.filtering.fit  
     (print.graph.trend.filtering.fit), 93  
 print.pttf\_operator  
     (print.graph.trend.filtering.fit), 93  
 print.slpl\_tf(model-summary), 79  
  
 print.slpl\_tf\_operator  
     (print.graph.trend.filtering.fit), 93  
 print.ssrhe.hessian.cv.fit  
     (print.graph.trend.filtering.fit), 93  
 print.ssrhe.hessian.fit  
     (print.graph.trend.filtering.fit), 93  
 print.ssrhe.hessian.gcv.fit  
     (print.graph.trend.filtering.fit), 93  
 print.ssrhe.hessian.l1.fit  
     (print.graph.trend.filtering.fit), 93  
 print.ssrhe.hessian.operator  
     (print.graph.trend.filtering.fit), 93  
 print.ssrhe.hessian.refit  
     (print.graph.trend.filtering.fit), 93  
 print.summary.geosmooth\_fit  
     (model-summary), 79  
 print.summary.harmonic\_smoother, 96, 123  
 print.synthetic\_dataset, 97  
 print.transported.graph.hessian.operator  
     (print.graph.trend.filtering.fit), 93  
 pttf.geometry, 40, 41, 97, 99, 100  
 pttf.operator, 39, 41, 99, 101  
 pttf.operator.filter.rows, 101  
  
 refit, 7, 8, 32, 71, 83, 102, 104–107, 109, 111, 115, 117  
 refit.lpl\_tf(refit.lpl\_tf), 103  
 refit.lpl\_tf, 102, 103  
 refit.malps, 102, 104  
 refit.metric.graph.lowpass  
     (refit.metric.graph.lowpass.fit), 106  
 refit.metric.graph.lowpass.fit, 103, 106  
 refit.slpl\_tf(refit.slpl\_tf), 107  
 refit.slpl\_tf, 102, 107  
 refit.ssrhe.hessian.fit, 103, 108  
 refit.ssrhe.hessian.l1.fit, 103, 109  
 refit.ssrhe.hessian.l1.regression  
     (refit.ssrhe.hessian.l1.fit),

- 109
- refit.ssrhe.hessian.regression  
(refit.ssrhe.hessian.fit), 108
- residuals(), 54
- residuals.chart\_kernel (model-values),  
81
- residuals.graph.trend.filtering.fit  
(model-values), 81
- residuals.local\_likelihood  
(model-values), 81
- residuals.lpl\_tf (model-values), 81
- residuals.lps (model-values), 81
- residuals.malps (model-values), 81
- residuals.metric.graph.lowpass.fit  
(model-values), 81
- residuals.metric.graph.lowpass.refit  
(model-values), 81
- residuals.ps\_lps (model-values), 81
- residuals.pttf.trend.filtering.fit  
(model-values), 81
- residuals.slpl\_tf (model-values), 81
- residuals.ssrhe.hessian.fit  
(model-values), 81
- residuals.ssrhe.hessian.l1.fit  
(model-values), 81
- residuals.ssrhe.hessian.refit  
(model-values), 81
- select.masked.graph.field, 111
- slpl\_tf.operator, 43, 112
- smoother.matrix, 27, 32, 70, 71, 83, 103,  
114, 116, 117
- smoother.matrix(), 68–70
- smoother.matrix.lps, 115, 115
- smoother.matrix.malps, 115, 117
- ssrhe.hessian.operator, 44, 48, 49, 118
- ssrhe.support.grid, 47, 53, 122
- summary.chart\_kernel (model-summary), 79
- summary.density\_fit (model-summary), 79
- summary.graph.trend.filtering.fit  
(model-summary), 79
- summary.harmonic\_smoother, 61, 95, 96,  
123
- summary.local\_likelihood  
(model-summary), 79
- summary.lpl\_tf (model-summary), 79
- summary.lps (model-summary), 79
- summary.malps (model-summary), 79
- summary.metric.graph.lowpass.fit  
(model-summary), 79
- summary.metric.graph.lowpass.refit  
(model-summary), 79
- summary.ps\_lps (model-summary), 79
- summary.pttf.trend.filtering.fit  
(model-summary), 79
- summary.slpl\_tf (model-summary), 79
- summary.ssrhe.hessian.fit  
(model-summary), 79
- summary.ssrhe.hessian.l1.fit  
(model-summary), 79
- summary.ssrhe.hessian.refit  
(model-summary), 79
- support.masked.graph.field, 124
- synthetic.dataset.checksum, 125
- synthetic.registry.ids, 125
- synthetic.registry.seed, 126
- synthetic.registry.spec, 126
- synthetic.response.bernoulli, 127
- synthetic.response.clustering.gaussian,  
128
- synthetic.response.gaussian, 128
- synthetic.response.heteroskedastic.gaussian,  
129
- synthetic.response.laplace.outlier,  
129
- synthetic.sampling.stratified, 130
- synthetic.spec, 4, 131
- synthetic.truth.gaussian.mixture, 132
- synthetic.truth.logit, 133
- synthetic.truth.named, 133
- synthetic.truth.occupation.mixture,  
134
- synthetic.truth.polynomial, 135
- transported.graph.hessian.operator,  
135
- validate.synthetic.dataset, 141