Package: rfastloess (via r-universe)

October 22, 2026

Title High-Performance LOESS Smoothing for R

Version 3.0.0

Description Provides high-performance LOESS smoothing (Locally Estimated Scatterplot Smoothing) using a 'Rust' backend. Supports various weight functions, robustness iterations, streaming/online processing, cross-validation, and uncertainty quantification. Applicable to genomic data, time series, and general-purpose smoothing tasks.

License MIT + file LICENSE | Apache License (== 2.0)

SystemRequirements 'Rust' programming language (via 'rustup' or system tools)

Encoding UTF-8

Depends R (>= 4.4.0)

Roxygen list(markdown = TRUE, roclets = c(` `namespace", ` `rd", ` `srr::srr_stats_roclet"))

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

BugReports https://github.com/thisisamirv/loess-project/issues

URL https://github.com/thisisamirv/loess-project, https://thisisamirv.github.io/loess-project/r/

Config/rextendr/version 0.5.0

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libclang-dev

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

Date/Publication 2026-10-22 02:59:53 UTC

RemoteUrl https://github.com/thisisamirv/loess-project

RemoteRef HEAD

RemoteSha fe706fe1d914faa6b7a81bfcf7d59a5c36dbd2ff

RemoteSubdir bindings/r

Contents

add_point . . . 2
cv_opts . . . 3
finalize . . . 3
fit . . . 4
intervals_opts . . . 5
Loess . . . 5
OnlineLoess . . . 8
plot.LoessResult . . . 11
predict.Loess . . . 11
predict_window . . . 13
print.Loess . . . 14
print.LoessResult . . . 14
print.OnlineLoess . . . 15
print.StreamingLoess . . . 16
process_chunk . . . 16
StreamingLoess . . . 17
window_diagnostics . . . 20

Index **21**

`add_point` *Add a single point to an online LOESS model*

Description

Add a single point to an online LOESS model

Usage

```
add_point(model, ...)

## S3 method for class 'OnlineLoess'
add_point(model, x, y, weight = 1, ...)
```

Arguments

`model`	An `OnlineLoess` object.
`...`	Must be empty.
`x`	A numeric coordinate vector with one value per configured dimension. For one-dimensional models, a scalar is also accepted.
`y`	A single numeric y value.
`weight`	Finite non-negative case weight for this observation; defaults to 1.

Value

An online result list, or `NULL` if fewer than `min_points` have been added.

Examples

```
model <- OnlineLoess(fraction = 0.2, window_capacity = 20L, min_points = 2L)
invisible(add_point(model, 1.0, 0.5))
add_point(model, 2.0, 0.6)
```

---

cv_opts *Cross-validation options for* Loess

Description

Cross-validation options for Loess

Usage

```
cv_opts(fractions, method = "kfold", k = 5L)
```

Arguments

fractions	Numeric vector of candidate smoothing fractions.
method	Cross-validation method: "kfold" or "loocv".
k	Number of folds for k-fold cross-validation. Default: 5.

Value

A cv_opts list for Loess(cv = ...).

Examples

```
model <- Loess(cv = cv_opts(fractions = c(0.2, 0.3, 0.5)))
```

---

finalize *Finalize a streaming LOESS model*

Description

Finalize a streaming LOESS model

Usage

```
finalize(model, ...)
```

Arguments

model	A StreamingLoess object.
...	Must be empty.

Value

A LoessResult combining all processed chunks.

Examples

```
x <- seq(0, 10, length.out = 100)
y <- sin(x) + rnorm(100, 0, 0.1)
model <- StreamingLoess(fraction = 0.2, chunk_size = 50L)
invisible(process_chunk(model, x[1:50], y[1:50]))
finalize(model)
```

---

fit *Fit a LOESS model to data*

Description

Fit a LOESS model to data

Usage

```
fit(model, ...)

## S3 method for class 'Loess'
fit(model, x, y, custom_weights = NULL, ...)
```

Arguments

model	A Loess object.
...	Must be empty.
x	Numeric vector of predictor values.
y	Numeric vector of response values.
custom_weights	Optional numeric vector of non-negative per-observation weights. NULL (default) applies no custom weighting.

Value

A LoessResult object.

Examples

```
x <- seq(0, 10, length.out = 100)
y <- sin(x) + rnorm(100, 0, 0.1)
model <- Loess(fraction = 0.2)
fit(model, x, y)
```

intervals_opts *Interval options for fitting and prediction*

Description

Interval options for fitting and prediction

Usage

```
intervals_opts(confidence = NULL, prediction = NULL)
```

Arguments

confidence	Confidence coverage level in (0, 1), or NULL.
prediction	Prediction coverage level in (0, 1), or NULL.

Value

A named list for the intervals argument.

Examples

```
model <- Loess(intervals = intervals_opts(confidence = 0.95))
```

Loess *LOESS Batch Smoothing*

Description

Create a stateful LOESS model for batch smoothing. This is the default mode: it processes the entire dataset at once and supports every feature (confidence/prediction intervals and cross-validation).

Usage

```
Loess(
    fraction = 0.67,
    ...,
    iterations = 3L,
    weight_function = "tricube",
    robustness_method = "bisquare",
    scaling_method = "mad",
    boundary_policy = "extend",
    outputs = NULL,
    intervals = NULL,
    zero_weight_fallback = "use_local_mean",
    auto_converge = NULL,
```

```
    parallel = TRUE,
    degree = "linear",
    dimensions = 1L,
    distance_metric = "normalized",
    surface_mode = "interpolation",
    weighted_metric_weights = NULL,
    cell = NULL,
    interpolation_vertices = NULL,
    boundary_degree_fallback = NULL,
    seed = NULL,
    missing = "error",
    retain_model = FALSE,
    cv = NULL
)
```

Arguments

`fraction`	Smoothing fraction, greater than 0 and up to 1. Default: 0.67. See Details for guidance on choosing a value.
`...`	Not used; forces all subsequent arguments to be named.
`iterations`	Number of robustness iterations, between 0 and 1000 (inclusive). Default: 3.
`weight_function`	Kernel weight function. One of `"tricube"` (default), `"gaussian"`, `"uniform"` (alias: `"boxcar"`), `"cosine"`, `"epanechnikov"`, `"biweight"` (alias: `"bisquare"`), or `"triangle"` (alias: `"triangular"`).
`robustness_method`	Outlier downweighting method: `"bisquare"` (default; alias: `"biweight"`), `"huber"`, or `"talwar"`.
`scaling_method`	Residual scale estimation for robustness weights: `"mad"` (default; alias: `"median_absolute_deviation"` `"mar"` (alias: `"median_absolute_residual"`), or `"mean"` (alias: `"mean_absolute_residual"`).
`boundary_policy`	Boundary handling strategy: `"extend"` (default; alias: `"pad"`), `"reflect"` (alias: `"mirror"`), `"zero"`, or `"noboundary"` (alias: `"none"`).
`outputs`	Optional character vector selecting `"diagnostics"`, `"residuals"`, `"weights"`, `"gradient"` (or `"derivative"`), `"se"`, and `"sorted"`. `NULL` (default) selects no optional components.
`intervals`	Grouped coverage levels from `intervals_opts`.
`zero_weight_fallback`	Fallback policy when all robustness weights drop to zero: `"use_local_mean"` (default; aliases: `"local_mean"`, `"mean"`), `"return_original"` (alias: `"original"`), or `"return_none"` (alias: `"none"`).
`auto_converge`	Convergence tolerance for early stopping of robustness iterations. `NULL` (default) disables early stopping.
`parallel`	Logical; enable parallel processing. Default: `TRUE`.
`degree`	Local polynomial degree: `"constant"`, `"linear"` (default), `"quadratic"`, `"cubic"`, or `"quartic"`.

`dimensions` Number of predictor dimensions. Default: 1.

`distance_metric`
Distance metric for neighbourhood computation: `"normalized"` (default), `"euclidean"`, `"manhattan"`, `"chebyshev"`, `"minkowski"`, or `"weighted"`. Use `"minkowski:p"` to set a custom p value.

`surface_mode` Surface evaluation mode: `"interpolation"` (default) or `"direct"`.

`weighted_metric_weights`
Numeric vector of per-dimension weights. Length must equal `dimensions`. Only used when `distance_metric = "weighted"`; setting `distance_metric = "weighted"` without providing this raises an error. `NULL` (default) has no effect unless `distance_metric = "weighted"` is set.

`cell` Cell size tuning parameter for the interpolation grid. `NULL` (default) uses the library default.

`interpolation_vertices`
Number of vertices in the interpolation grid. `NULL` (default) uses the library default.

`boundary_degree_fallback`
Logical; if `TRUE`, interpolation vertices lying outside the range of the data are fitted with a linear model rather than the requested degree, avoiding unstable extrapolation. It has no effect below quadratic degree, and `FALSE` reproduces `stats::loess()`. `NULL` (default) uses the library default.

`seed` Seed for reproducible CV folds, or `NULL`.

`missing` Policy for non-finite (NaN/Inf) values in the input data: `"error"` (default) raises an error, `"drop"` silently removes observations (rows) where any x dimension or y is non-finite (and the matching `custom_weights` entry) before fitting. A length mismatch between x and y always raises an error, even under `"drop"`.

`retain_model` Logical; if `TRUE`, retain the fitted model's training data, enabling `predict.Loess` for out-of-sample prediction. Default: `FALSE`.

`cv` Grouped cross-validation settings from `cv_opts`.

Details

Best suited when the dataset fits in memory and you need intervals, cross-validation, or diagnostics. For datasets that don't fit in memory or arrive in chunks, see `StreamingLoess`; for point-by-point real-time data, see `OnlineLoess`.

`fraction` is the most important parameter: it controls the size of the local neighbourhood used at each point.

When `outputs` includes `"diagnostics"`, Batch `residual_sd` is the robust residual scale estimate `1.4826 * MAD`.

Range	Effect	Use case
0.1-0.3	Fine detail	Rapidly changing signals
0.3-0.5	Balanced	General purpose
0.5-0.7	Heavy smoothing	Noisy data
0.7-1.0	Very smooth	Trend extraction

Value

A Loess object.

Examples

```
x <- seq(0, 10, length.out = 100)
y <- sin(x) + rnorm(100, 0, 0.1)
model <- Loess(fraction = 0.2)
result <- fit(model, x, y)
plot(x, y)
lines(x, result$y, col = "red")
```

`OnlineLoess` *LOESS Online Smoothing*

Description

Create a stateful LOESS model for real-time online data. Maintains a sliding window and processes each incoming point immediately via `add_point`.

Usage

```
OnlineLoess(
    fraction = 0.67,
    window_capacity = 1000L,
    min_points = 2L,
    ...,
    iterations = 0L,
    weight_function = "tricube",
    robustness_method = "bisquare",
    scaling_method = "mad",
    boundary_policy = "extend",
    zero_weight_fallback = "use_local_mean",
    update_mode = "incremental",
    auto_converge = NULL,
    intervals = NULL,
    degree = "linear",
    dimensions = 1L,
    distance_metric = "normalized",
    surface_mode = "interpolation",
    weighted_metric_weights = NULL,
    cell = NULL,
    interpolation_vertices = NULL,
    boundary_degree_fallback = NULL,
```

```
    missing = "error",
    outputs = NULL
)
```

Arguments

`fraction`	Smoothing fraction, greater than 0 and up to 1. Default: 0.67. See Details for guidance on choosing a value.
`window_capacity`	Maximum number of points kept in the sliding window, at least 3. Default: 1000.
`min_points`	Minimum number of points required before smoothing begins, between 2 and `window_capacity`. Default: 2.
`...`	Not used; forces all subsequent arguments to be named.
`iterations`	Number of robustness iterations, between 0 and 1000 (inclusive). Default: 3.
`weight_function`	Kernel weight function. One of `"tricube"` (default), `"gaussian"`, `"uniform"` (alias: `"boxcar"`), `"cosine"`, `"epanechnikov"`, `"biweight"` (alias: `"bisquare"`), or `"triangle"` (alias: `"triangular"`).
`robustness_method`	Outlier downweighting method: `"bisquare"` (default; alias: `"biweight"`), `"huber"`, or `"talwar"`.
`scaling_method`	Residual scale estimation for robustness weights: `"mad"` (default; alias: `"median_absolute_deviation"` `"mar"` (alias: `"median_absolute_residual"`), or `"mean"` (alias: `"mean_absolute_residual"`).
`boundary_policy`	Boundary handling strategy: `"extend"` (default; alias: `"pad"`), `"reflect"` (alias: `"mirror"`), `"zero"`, or `"noboundary"` (alias: `"none"`).
`zero_weight_fallback`	Fallback policy when all robustness weights drop to zero: `"use_local_mean"` (default; aliases: `"local_mean"`, `"mean"`), `"return_original"` (alias: `"original"`), or `"return_none"` (alias: `"none"`).
`update_mode`	Window update strategy: `"incremental"` (default; alias: `"single"`) updates only the newest point; `"full"` (alias: `"resmooth"`) re-smooths all window points after each addition.
`auto_converge`	Convergence tolerance for early stopping of robustness iterations. `NULL` (default) disables early stopping.
`intervals`	Grouped coverage levels from `intervals_opts`.
`degree`	Local polynomial degree: `"constant"`, `"linear"` (default), `"quadratic"`, `"cubic"`, or `"quartic"`.
`dimensions`	Number of predictor dimensions; each `add_point()` call must provide one x coordinate per dimension.
`distance_metric`	Distance metric for neighbourhood computation: `"normalized"` (default), `"euclidean"`, `"manhattan"`, `"chebyshev"`, `"minkowski"`, or `"weighted"`. Use `"minkowski:p"` to set a custom p value.

surface_mode — Surface evaluation mode: "interpolation" (default) or "direct".

weighted_metric_weights — Numeric vector of per-dimension weights. Length must equal dimensions. Only used when distance_metric = "weighted"; setting distance_metric = "weighted" without providing this raises an error. NULL (default) has no effect unless distance_metric = "weighted" is set.

cell — Cell size tuning parameter for the interpolation grid. NULL (default) uses the library default.

interpolation_vertices — Number of vertices in the interpolation grid. NULL (default) uses the library default.

boundary_degree_fallback — Logical; if TRUE, interpolation vertices lying outside the range of the data are fitted with a linear model rather than the requested degree, avoiding unstable extrapolation. It has no effect below quadratic degree, and FALSE reproduces stats::loess(). NULL (default) uses the library default.

missing — Policy for a non-finite (NaN/Inf) x or y value passed to add_point: "error" (default) raises an error, "drop" silently ignores the point (returns NULL) instead of adding it to the window.

outputs — Optional character vector selecting "weights", "gradient" (or "derivative"), and "se". NULL (default) selects no optional components; "se" requires full update mode.

Details

Best suited when data arrives incrementally (e.g. sensors or streams), real-time smoothed values are needed, or memory is fixed. For datasets that fit in memory, see Loess; for large batches processed in chunks, see StreamingLoess.

Value

An OnlineLoess object.

Examples

```
model <- OnlineLoess(fraction = 0.2, window_capacity = 20)
x <- 1:50
y <- sin(x * 0.1) + rnorm(50, 0, 0.1)
smoothed <- numeric(0)
for (i in seq_along(x)) {
    result <- add_point(model, x[i], y[i])
    if (!is.null(result)) smoothed <- c(smoothed, result$y)
}
head(smoothed, 5)
```

plot.LoessResult *Plot Loess Result*

Description

Plotting supports one-dimensional results. For a multivariate fit, select a predictor dimension and create a custom plot.

Usage

```
## S3 method for class 'LoessResult'
plot(x, main = "LOESS Fit", ...)
```

Arguments

x	A LoessResult object.
main	Plot title.
...	Additional arguments passed to plot() and lines().

Value

NULL, invisibly. Called for side effects (plotting).

Examples

```
x <- seq(0, 10, length.out = 100)
y <- sin(x) + rnorm(100, 0, 0.1)
model <- Loess(fraction = 0.2)
res <- fit(model, x, y)
plot(res)
```

predict.Loess *Predict from a fitted LOESS model at out-of-sample points*

Description

Predict from a fitted LOESS model at out-of-sample points

Usage

```
## S3 method for class 'Loess'
predict(
    object,
    new_x,
    intervals = NULL,
    extrapolation = "clamp",
    max_extrapolation_distance = NULL,
    max_neighbor_distance = NULL,
    outputs = NULL,
    ...
)
```

Arguments

object	A Loess object, fitted (via fit) with retain_model = TRUE passed to Loess.
new_x	Numeric vector of out-of-sample query points (flattened, dimensions values per point).
intervals	Grouped coverage levels from intervals_opts.
extrapolation	Behavior for query points outside the training range: "clamp" (default), "linear", or "error".
max_extrapolation_distance	Under "linear" extrapolation, the maximum allowed distance beyond the training boundary before predict errors instead of returning an unbounded value. NULL (default) disables the cap.
max_neighbor_distance	Maximum allowed distance to the farthest point in a query's neighbor window before predict errors, catching in-range-but-sparse query points. NULL (default) disables the cap.
outputs	Optional character vector selecting "se", "gradient", or "derivative". NULL (default) selects no optional components.
...	Must be empty.

Value

A list with a y element (predicted values) and optional standard_errors/confidence_lower/confidence_upper/prediction_lower/prediction_upper/derivative elements.

Examples

```
x <- seq(0, 10, length.out = 100)
y <- sin(x) + rnorm(100, 0, 0.1)
model <- Loess(fraction = 0.2, retain_model = TRUE)
fit(model, x, y)
predict(model, c(2.5, 7.5))
```

predict_window *Predict from the current Online window*

Description

Predict from the current Online window

Usage

```
predict_window(model, ...)

## S3 method for class 'OnlineLoess'
predict_window(
    model,
    new_x,
    outputs = NULL,
    intervals = NULL,
    extrapolation = "clamp",
    max_extrapolation_distance = NULL,
    max_neighbor_distance = NULL,
    ...
)
```

Arguments

model	An OnlineLoess object.
...	Must be empty.
new_x	Numeric query points (flattened, one coordinate per dimension).
outputs	Optional character vector selecting "se" and/or "gradient" (alias "derivative").
intervals	Grouped coverage levels from intervals_opts.
extrapolation	Behavior outside the window's predictor bounds: "clamp" (default), "linear", or "error".
max_extrapolation_distance	Optional cap for linear extrapolation.
max_neighbor_distance	Optional cap for sparse-neighborhood predictions.

Value

A list containing predicted values and requested optional outputs.

Examples

```
model <- OnlineLoess(
    fraction = 1, window_capacity = 20L, surface_mode = "direct"
)
for (x in 1:10) {
    invisible(add_point(model, x, 2 * x + 1))
}
predict_window(model, c(4.5, 5.5), outputs = "gradient")
```

---

print.Loess *Print Loess Model*

Description

Print Loess Model

Usage

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

Arguments

x	A Loess object.
...	Additional arguments (ignored).

Value

The input object x, invisibly.

Examples

```
model <- Loess(fraction = 0.3)
print(model)
```

---

print.LoessResult *Print Loess Result*

Description

Print Loess Result

Usage

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

Arguments

x	A LoessResult object.
...	Additional arguments (ignored).

Value

The input object x, invisibly.

Examples

```
x <- seq(0, 10, length.out = 50)
y <- sin(x) + rnorm(50, 0, 0.1)
model <- Loess(fraction = 0.3)
result <- fit(model, x, y)
print(result)
```

---

print.OnlineLoess *Print OnlineLoess Model*

Description

Print OnlineLoess Model

Usage

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

Arguments

x	An OnlineLoess object.
...	Additional arguments.

Value

The input object x, invisibly.

Examples

```
model <- OnlineLoess(fraction = 0.2, window_capacity = 20L)
print(model)
```

print.StreamingLoess *Print StreamingLoess Model*

Description

Print StreamingLoess Model

Usage

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

Arguments

x	A StreamingLoess object.
...	Additional arguments.

Value

The input object x, invisibly.

Examples

```
model <- StreamingLoess(fraction = 0.3, chunk_size = 50L)
print(model)
```

process_chunk *Process a data chunk through a streaming LOESS model*

Description

Process a data chunk through a streaming LOESS model

Usage

```
process_chunk(model, ...)

## S3 method for class 'StreamingLoess'
process_chunk(model, x, y, custom_weights = NULL, ...)
```

Arguments

model	A StreamingLoess object.
...	Must be empty.
x	Numeric vector of x values.
y	Numeric vector of y values.
custom_weights	Optional numeric case weight per observation.

Value

A LoessResult for this chunk.

Examples

```
x <- seq(0, 10, length.out = 100)
y <- sin(x) + rnorm(100, 0, 0.1)
model <- StreamingLoess(fraction = 0.2, chunk_size = 50L)
process_chunk(model, x[1:50], y[1:50])
```

StreamingLoess *LOESS Streaming Smoothing*

Description

Create a stateful LOESS model for streaming data. Processes data in fixed-size chunks with configurable overlap: results for each chunk are returned by process_chunk, and finalize flushes any remaining buffered points after the last chunk.

Usage

```
StreamingLoess(
    fraction = 0.67,
    chunk_size = 5000L,
    ...,
    overlap = NULL,
    iterations = 3L,
    weight_function = "tricube",
    robustness_method = "bisquare",
    scaling_method = "mad",
    boundary_policy = "extend",
    zero_weight_fallback = "use_local_mean",
    auto_converge = NULL,
    intervals = NULL,
    merge_strategy = "weighted_average",
    parallel = TRUE,
    degree = "linear",
    dimensions = 1L,
    distance_metric = "normalized",
    surface_mode = "interpolation",
    weighted_metric_weights = NULL,
    cell = NULL,
    interpolation_vertices = NULL,
    boundary_degree_fallback = NULL,
    missing = "error",
    outputs = NULL
)
```

Arguments

fraction	Smoothing fraction, greater than 0 and up to 1. Default: 0.67. See Details for guidance on choosing a value.
chunk_size	Number of data points per processing chunk, at least 10. Default: 5000.
...	Not used; forces all subsequent arguments to be named.
overlap	Number of overlapping points between consecutive chunks, less than chunk_size. NULL (default) uses chunk_size / 10 (clamped to [1, chunk_size - 10]).
iterations	Number of robustness iterations, between 0 and 1000 (inclusive). Default: 3.
weight_function	Kernel weight function. One of "tricube" (default), "gaussian", "uniform" (alias: "boxcar"), "cosine", "epanechnikov", "biweight" (alias: "bisquare"), or "triangle" (alias: "triangular").
robustness_method	Outlier downweighting method: "bisquare" (default; alias: "biweight"), "huber", or "talwar".
scaling_method	Residual scale estimation for robustness weights: "mad" (default; alias: "median_absolute_deviation" "mar" (alias: "median_absolute_residual"), or "mean" (alias: "mean_absolute_residual").
boundary_policy	Boundary handling strategy: "extend" (default; alias: "pad"), "reflect" (alias: "mirror"), "zero", or "noboundary" (alias: "none").
zero_weight_fallback	Fallback policy when all robustness weights drop to zero: "use_local_mean" (default; aliases: "local_mean", "mean"), "return_original" (alias: "original"), or "return_none" (alias: "none").
auto_converge	Convergence tolerance for early stopping of robustness iterations. NULL (default) disables early stopping.
intervals	Grouped coverage levels from intervals_opts.
merge_strategy	Strategy for reconciling overlapping chunk regions: "weighted_average" (default; alias: "weighted"), "average" (alias: "mean"), "take_first" (alias: "first"), or "take_last" (alias: "last").
parallel	Logical; enable parallel processing. Default: TRUE.
degree	Local polynomial degree: "constant", "linear" (default), "quadratic", "cubic", or "quartic".
dimensions	Number of predictor dimensions. Default: 1.
distance_metric	Distance metric for neighbourhood computation: "normalized" (default), "euclidean", "manhattan", "chebyshev", "minkowski", or "weighted". Use "minkowski:p" to set a custom p value.
surface_mode	Surface evaluation mode: "interpolation" (default) or "direct".
weighted_metric_weights	Numeric vector of per-dimension weights. Length must equal dimensions. Only used when distance_metric = "weighted"; setting distance_metric = "weighted" without providing this raises an error. NULL (default) has no effect unless distance_metric = "weighted" is set.

`cell` Cell size tuning parameter for the interpolation grid. `NULL` (default) uses the library default.

`interpolation_vertices` Number of vertices in the interpolation grid. `NULL` (default) uses the library default.

`boundary_degree_fallback` Logical; if `TRUE`, interpolation vertices lying outside the range of the data are fitted with a linear model rather than the requested degree, avoiding unstable extrapolation. It has no effect below quadratic degree, and `FALSE` reproduces `stats::loess()`. `NULL` (default) uses the library default.

`missing` Policy for non-finite (NaN/Inf) values in the input data: `"error"` (default) raises an error, `"drop"` silently removes observations (rows) where any x dimension or y is non-finite (and the matching `custom_weights` entry) before fitting. A length mismatch between x and y always raises an error, even under `"drop"`.

`outputs` Optional character vector selecting `"diagnostics"`, `"residuals"`, `"weights"`, `"gradient"` (or `"derivative"`), and `"se"`. `NULL` (default) selects no optional components.

Details

When `outputs` includes `"diagnostics"`, `residual_sd` is the cumulative sample standard deviation of emitted residuals. Batch `residual_sd` instead uses the robust residual scale estimate `1.4826 * MAD`. Best suited for datasets over 100,000 points, memory-constrained environments, or batch processing pipelines. For smaller datasets that fit in memory, see `Loess`; for point-by-point real-time data, see `OnlineLoess`.

Overlapping regions between chunks are reconciled via `merge_strategy`:

Strategy	Behavior
`"average"`	Arithmetic mean of both estimates
`"weighted_average"`	Distance-weighted blend (default)
`"take_first"`	Keep left-chunk estimate
`"take_last"`	Keep right-chunk estimate

Value

A StreamingLoess object.

Examples

```
x <- seq(0, 10, length.out = 100)
y <- sin(x) + rnorm(100, 0, 0.1)
model <- StreamingLoess(fraction = 0.2, chunk_size = 50)
res1 <- process_chunk(model, x[1:50], y[1:50])
res2 <- process_chunk(model, x[51:100], y[51:100])
finalize(model)
```

`window_diagnostics` *Compute diagnostics for the current Online window*

Description

Compute diagnostics for the current Online window

Usage

```
window_diagnostics(model, ...)
```

Arguments

`model`	An OnlineLoess object.
`...`	Must be empty.

Value

A list of goodness-of-fit metrics, or `NULL` until the window reaches `min_points`.

Examples

```
model <- OnlineLoess(fraction = 1, window_capacity = 20L)
for (x in 1:10) {
    invisible(add_point(model, x, 2 * x + 1))
}
window_diagnostics(model)
```

Index

add_point, 2, *8*, *10*

cv_opts, 3, *7*

finalize, 3, *17*
fit, 4, *12*

intervals_opts, 5, *6*, *9*, *12*, *13*, *18*

Loess, *3*, 5, *10*, *12*, *19*

OnlineLoess, *7*, 8, *19*

plot.LoessResult, 11
predict.Loess, *7*, 11
predict_window, 13
print.Loess, 14
print.LoessResult, 14
print.OnlineLoess, 15
print.StreamingLoess, 16
process_chunk, 16, *17*

StreamingLoess, *7*, *10*, 17

window_diagnostics, 20