

Package: rfastlowess (via r-universe)

October 12, 2026

Title High-Performance LOWESS Smoothing for R

Version 5.0.0

Description Provides high-performance LOWESS smoothing (Locally Weighted 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)

Imports digest, jsonlite

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/lowess-project/issues>

URL <https://github.com/thisisamirv/lowess-project>,
<https://thisisamirv.github.io/lowess-project/r/>

Config/rextendr/version 0.5.0

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libclang-dev

Repository <https://ropensci-staging.r-universe.dev>

Date/Publication 2026-10-12 17:46:36 UTC

RemoteUrl <https://github.com/thisisamirv/lowess-project>

RemoteRef HEAD

RemoteSha ef3ffbe373656c1ee6acd74505d8fc472830b502

RemoteSubdir bindings/r

Contents

add_point	2
cv_opts	3
finalize	4
fit	4
gpu_available	5
install_gpu	6
intervals_opts	7
Lowess	7
OnlineLowess	10
plot.LowessResult	12
predict.Lowess	13
print.Lowess	14
print.LowessResult	15
print.OnlineLowess	15
print.StreamingLowess	16
process_chunk	16
StreamingLowess	17
Index	20

add_point	<i>Add a single point to an online LOWESS model</i>
-----------	---

Description

Add a single point to an online LOWESS model

Usage

```
add_point(model, ...)  
  
## S3 method for class 'OnlineLowess'  
add_point(model, x, y, ...)
```

Arguments

- model An OnlineLowess object.
- ... Must be empty.
- x A single numeric x value.
- y A single numeric y value.

Value

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

Examples

```
model <- OnlineLowess(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 [Lowess](#)***Description**

Build a cross-validation options list to pass to `Lowess(cv = ...)`. Cross-validation is disabled by default (`cv = NULL`); supplying `cv_opts()` enables it and lets the model pick the best smoothing fraction from the candidates.

Usage

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

Arguments

method	Cross-validation method: "kfold" (default) or "loocv".
k	Number of folds for k-fold cross-validation. Default: 5.
fractions	Numeric vector of candidate smoothing fractions, each greater than 0 and up to 1 (e.g. <code>c(0.2, 0.3, 0.5)</code>).

Value

A `cv_opts` list for `Lowess(cv = ...)`. Use `Lowess(seed = ...)` for reproducible fold assignment.

Examples

```
model <- Lowess(
  cv = cv_opts(method = "kfold", k = 5, fractions = c(0.2, 0.3, 0.5)),
  seed = 42
)
```

finalize	<i>Finalize a streaming LOWESS model</i>
----------	--

Description

Finalize a streaming LOWESS model

Usage

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

Arguments

model	A StreamingLowess object.
...	Must be empty.

Value

A LowessResult combining all processed chunks.

Examples

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

fit	<i>Fit a LOWESS model to data</i>
-----	-----------------------------------

Description

Fit a LOWESS model to data

Usage

```
fit(model, ...)

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

Arguments

model	A Lowess 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 LowessResult object.

Examples

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

gpu_available

Check GPU Backend Availability

Description

Returns whether the currently loaded **rfastlowess** shared library was built with the GPU backend enabled.

Usage

```
gpu_available()
```

Value

Logical; TRUE if the GPU backend is active.

See Also

[install_gpu](#) to download and install it.

Examples

```
gpu_available()
```

install_gpu

*Download and Install the GPU-Enabled Backend***Description**

Downloads a prebuilt GPU-enabled **rfastlowess** shared library for the current platform from the matching GitHub Release. On Windows, installs a versioned sidecar alongside the CPU library and activates its native routines when the package is loaded after restarting R. Other platforms atomically replace the CPU library. GPU support is opt-in and not included in CRAN/Bioconductor releases.

A running R session cannot swap an already-loaded shared library, so **restart R** after installing for the change to take effect.

Downloads are checked against GitHub's published SHA-256 digest before execution. Native probes have a 30-second timeout and require matching package-version/ABI contracts and registered argument counts. Older GPU artifacts without this contract must be rebuilt. On Windows, invalid sidecars are skipped and activated DLLs remain mapped until R exits to protect live model finalizers. Probing is not a security sandbox; local native libraries must be trusted.

Usage

```
install_gpu(yes = FALSE, local_path = NULL)
```

Arguments

yes	Logical; skip the interactive y/N confirmation prompt. Must be TRUE when the session is not interactive.
local_path	Character; path to a GPU-enabled shared library already built locally (e.g. via WITH_GPU=1 make install in benchmarks/). When given, skips the GitHub Release lookup/download and installs directly from this path — useful for testing the installer itself, or installing an unreleased build.

Value

Invisibly, TRUE if a GPU-enabled library is available at the printed path (already active, or freshly installed); FALSE if the user aborted.

See Also

[gpu_available](#) to check the current status.

Examples

```
# Check whether the GPU backend is already active before installing it
gpu_available()
if (interactive()) {
  install_gpu()
}
```

intervals_opts	<i>Interval options for LOWESS models</i>
----------------	---

Description

Build an interval options list to pass to `intervals = ...` in [Lowess](#), [StreamingLowess](#), [OnlineLowess](#), or [predict.Lowess](#).

Usage

```
intervals_opts(confidence = NULL, prediction = NULL, bootstrap = 0L)
```

Arguments

confidence	Coverage for confidence intervals, greater than 0 and less than 1 (e.g. 0.90). NULL (default) disables them.
prediction	Coverage for prediction intervals, greater than 0 and less than 1 (e.g. 0.99). NULL (default) disables them. This level is independent of confidence.
bootstrap	Number of bootstrap resamples used to compute the intervals. 0 (default) uses the analytic intervals.

Value

An `intervals_opts` list.

Examples

```
model <- Lowess(
  intervals = intervals_opts(
    confidence = 0.90, prediction = 0.99, bootstrap = 200
  ),
  seed = 42
)
```

Lowess	<i>LOWESS Batch Smoothing</i>
--------	-------------------------------

Description

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

Usage

```
Lowess(
  fraction = 0.67,
  ...,
  iterations = 3L,
  weight_function = "tricube",
  robustness_method = "bisquare",
  delta = NULL,
  zero_weight_fallback = "use_local_mean",
  boundary_policy = "extend",
  scaling_method = "mad",
  auto_converge = NULL,
  missing = "error",
  parallel = TRUE,
  backend = "cpu",
  outputs = NULL,
  intervals = NULL,
  cv = NULL,
  seed = NULL,
  retain_model = FALSE
)
```

Arguments

<code>fraction</code>	Smoothing fraction, greater than 0 and up to 1. Default: 0.67.
<code>...</code>	Not used; forces all subsequent arguments to be named.
<code>iterations</code>	Number of robustness iterations, between 0 and 1000 (inclusive). Default: 3.
<code>weight_function</code>	Kernel weight function. One of "tricube" (default), "gaussian", "uniform" (alias: "boxcar"), "cosine", "epanechnikov", "biweight" (alias: "bisquare"), or "triangle" (alias: "triangular").
<code>robustness_method</code>	Outlier downweighting method: "bisquare" (default; alias: "biweight"), "huber", or "talwar".
<code>delta</code>	Interpolation distance threshold, as a non-negative fraction of the x range; points within delta of each other on x share the same local fit. NULL (default) sets it automatically to 1/100th of the x range.
<code>zero_weight_fallback</code>	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").
<code>boundary_policy</code>	Boundary handling strategy: "extend" (default; alias: "pad"), "reflect" (alias: "mirror"), "zero", or "noboundary" (alias: "none").
<code>scaling_method</code>	Residual scale estimation for robustness weights: "mad" (default; alias: "median_absolute_deviation", "mar" (alias: "median_absolute_residual"), or "mean" (alias: "mean_absolute_residual").

<code>auto_converge</code>	Convergence tolerance for early stopping of robustness iterations. NULL (default) disables early stopping.
<code>missing</code>	Policy for non-finite (NaN/Infinity) values in input data: "error" (default) raises an error, "drop" silently removes affected observations before fitting.
<code>parallel</code>	Logical; enable parallel processing. Default: TRUE.
<code>backend</code>	Execution backend: "cpu" (default) or "gpu". GPU support requires the package to be built locally with <code>WITH_GPU=1</code> (see <code>bindings/r/Makefile</code>) and a Vulkan/Metal/DX12-capable GPU driver; not available in released CRAN/Bioconductor binaries.
<code>outputs</code>	Character vector selecting optional output components: "se" (standard errors), "diagnostics", "residuals", "weights" (robustness weights), "derivative", and/or "sorted". NULL (default) returns only the core result (x, y, and fit meta-data).
<code>intervals</code>	Interval options, created with <code>intervals_opts</code> (or a named list with any of confidence, prediction, bootstrap): e.g. <code>intervals = intervals_opts(confidence = 0.90, prediction = 0.99, bootstrap = 200)</code> . Confidence and prediction coverage levels are independent and may differ. NULL (default) disables intervals.
<code>cv</code>	Cross-validation options, created with <code>cv_opts</code> : e.g. <code>cv = cv_opts(method = "kfold", k = 5, fractions = c(0.2, 0.3, 0.5))</code> . NULL (default) disables cross-validation.
<code>seed</code>	Non-negative whole-number seed shared by cross-validation and bootstrap resampling for reproducible results. When cross-validation is unavailable, it controls bootstrap resampling only. NULL (default) uses a random seed.
<code>retain_model</code>	Logical; if TRUE, retain the fitted model's training data, enabling <code>predict.Lowess</code> for out-of-sample prediction. Default: FALSE.

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 [StreamingLowess](#); for point-by-point real-time data, see [OnlineLowess](#). When "diagnostics" is selected, `residual_sd` is the robust residual scale estimate ($1.4826 * MAD$).

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

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 Lowess object.

Examples

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

OnlineLowess

LOWESS Online Smoothing

Description

Create a stateful LOWESS model for real-time online data. Maintains a sliding window and processes each incoming point immediately via [add_point](#).

Usage

```
OnlineLowess(
  fraction = 0.67,
  ...,
  iterations = 0L,
  weight_function = "tricube",
  robustness_method = "bisquare",
  delta = NULL,
  zero_weight_fallback = "use_local_mean",
  boundary_policy = "extend",
  scaling_method = "mad",
  auto_converge = NULL,
  missing = "error",
  window_capacity = 1000L,
  min_points = 2L,
  update_mode = "incremental",
  outputs = NULL,
  intervals = NULL,
  seed = NULL
)
```

Arguments

<code>fraction</code>	Smoothing fraction, greater than 0 and up to 1. Default: 0.67.
<code>...</code>	Not used; forces all subsequent arguments to be named.
<code>iterations</code>	Number of robustness iterations. Requires <code>update_mode = "full"</code> ; the default <code>"incremental"</code> mode is a non-robust single-point fit that ignores robustness iterations. Default: 0.

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".
delta	Interpolation distance as a non-negative fraction of the x range. In Online mode, NULL (default) disables interpolation; positive values require update_mode = "full".
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").
boundary_policy	Boundary handling strategy: "extend" (default; alias: "pad"), "reflect" (alias: "mirror"), "zero", or "noboundary" (alias: "none").
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").
auto_converge	Convergence tolerance for early stopping. Requires update_mode = "full" and iterations > 0; NULL (default) disables early stopping.
missing	Policy for non-finite (NaN/Infinity) values in input data: "error" (default) raises an error, "drop" silently removes affected observations before fitting.
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.
update_mode	Window update strategy: "incremental" (default; alias: "single") updates only the newest point and does not run robustness iterations; "full" (alias: "resmooth") re-smooths all window points after each addition and supports robustness iterations, positive delta, and auto_converge.
outputs	Character vector selecting optional Online output components: "se" (standard errors), "weights" (robustness weights), and/or "derivative". NULL (default) returns only the core result.
intervals	Interval options, created with intervals_opts (or a named list with any of confidence, prediction, bootstrap): e.g. intervals = intervals_opts(confidence = 0.90, prediction = 0.99, bootstrap = 200). Confidence and prediction coverage levels are independent and may differ. NULL (default) disables intervals.
seed	Non-negative whole-number seed shared by cross-validation and bootstrap re-sampling for reproducible results. When cross-validation is unavailable, it controls bootstrap resampling only. NULL (default) uses a random seed.

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 [Lowess](#); for large batches processed in chunks, see [StreamingLowess](#).

Confidence and prediction interval coverage levels are independent; both interval types require `update_mode = "full"`. Positive delta values also require `update_mode = "full"`. A non-NULL `auto_converge` requires full mode and at least one robustness iteration; unsupported combinations are rejected.

Value

An OnlineLowess object.

Examples

```
model <- OnlineLowess(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.LowessResult	<i>Plot Lowess Result</i>
-------------------	---------------------------

Description

Plot Lowess Result

Usage

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

Arguments

<code>x</code>	A LowessResult object.
<code>main</code>	Plot title.
<code>...</code>	Additional arguments passed to <code>plot()</code> and <code>lines()</code> .

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 <- Lowess(fraction = 0.2)
res <- fit(model, x, y)
plot(res)
```

predict.Lowess	<i>Predict from a fitted LOWESS model at out-of-sample points</i>
----------------	---

Description

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

Usage

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

Arguments

object	A Lowess object, fitted (via fit) with retain_model = TRUE passed to Lowess .
new_x	Numeric vector of out-of-sample query points.
outputs	Character vector selecting optional prediction components: "se" and/or "derivative". NULL (default) returns only predicted values.
intervals	Interval options, created with intervals_opts (or a named list with any of confidence, prediction, bootstrap). Confidence and prediction coverage levels are independent; for example, intervals_opts(confidence = 0.90, prediction = 0.99). NULL (default) disables intervals.
seed	Non-negative whole-number seed for bootstrap resampling. NULL (default) uses a random seed.
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 local window before predict errors, catching in-range-but-sparse query points. NULL (default) disables the cap.

`...`
 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 <- Lowess(fraction = 0.2, retain_model = TRUE)
fit(model, x, y)
predict(model, c(2.5, 7.5))
```

`print.Lowess`

Print Lowess Model

Description

Print Lowess Model

Usage

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

Arguments

`x` A Lowess object.

`...` Additional arguments (ignored).

Value

The input object `x`, invisibly.

Examples

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

print.LowessResult	<i>Print Lowess Result</i>
--------------------	----------------------------

Description

Print Lowess Result

Usage

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

Arguments

x	A LowessResult 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 <- Lowess(fraction = 0.3)  
result <- fit(model, x, y)  
print(result)
```

print.OnlineLowess	<i>Print OnlineLowess Model</i>
--------------------	---------------------------------

Description

Print OnlineLowess Model

Usage

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

Arguments

x	An OnlineLowess object.
...	Additional arguments.

Value

The input object x, invisibly.

Examples

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

```
print.StreamingLowess Print StreamingLowess Model
```

Description

Print StreamingLowess Model

Usage

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

Arguments

x A StreamingLowess object.
... Additional arguments.

Value

The input object x, invisibly.

Examples

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

```
process_chunk                    Process a data chunk through a streaming LOWESS model
```

Description

Process a data chunk through a streaming LOWESS model

Usage

```
process_chunk(model, ...)

## S3 method for class 'StreamingLowess'
process_chunk(model, x, y, ...)
```

Arguments

<code>model</code>	A StreamingLowess object.
<code>...</code>	Must be empty.
<code>x</code>	Numeric vector of x values.
<code>y</code>	Numeric vector of y values.

Value

A `LowessResult` for this chunk.

Examples

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

StreamingLowess	<i>LOWESS Streaming Smoothing</i>
-----------------	-----------------------------------

Description

Create a stateful LOWESS 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

```
StreamingLowess(
  fraction = 0.67,
  ...,
  iterations = 3L,
  weight_function = "tricube",
  robustness_method = "bisquare",
  delta = NULL,
  zero_weight_fallback = "use_local_mean",
  boundary_policy = "extend",
  scaling_method = "mad",
  auto_converge = NULL,
  missing = "error",
  chunk_size = 5000L,
  overlap = NULL,
  merge_strategy = "weighted_average",
  parallel = TRUE,
  outputs = NULL,
  intervals = NULL,
  seed = NULL
)
```

Arguments

<code>fraction</code>	Smoothing fraction, greater than 0 and up to 1. Default: 0.67.
<code>...</code>	Not used; forces all subsequent arguments to be named.
<code>iterations</code>	Number of robustness iterations, between 0 and 1000 (inclusive). Default: 3.
<code>weight_function</code>	Kernel weight function. One of "tricube" (default), "gaussian", "uniform" (alias: "boxcar"), "cosine", "epanechnikov", "biweight" (alias: "bisquare"), or "triangle" (alias: "triangular").
<code>robustness_method</code>	Outlier downweighting method: "bisquare" (default; alias: "biweight"), "huber", or "talwar".
<code>delta</code>	Interpolation distance threshold, as a non-negative fraction of the x range; points within delta of each other on x share the same local fit. NULL (default) sets it automatically to 1/100th of the x range.
<code>zero_weight_fallback</code>	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").
<code>boundary_policy</code>	Boundary handling strategy: "extend" (default; alias: "pad"), "reflect" (alias: "mirror"), "zero", or "noboundary" (alias: "none").
<code>scaling_method</code>	Residual scale estimation for robustness weights: "mad" (default; alias: "median_absolute_deviation"), "mar" (alias: "median_absolute_residual"), or "mean" (alias: "mean_absolute_residual").
<code>auto_converge</code>	Convergence tolerance for early stopping of robustness iterations. NULL (default) disables early stopping.
<code>missing</code>	Policy for non-finite (NaN/Infinity) values in input data: "error" (default) raises an error, "drop" silently removes affected observations before fitting.
<code>chunk_size</code>	Number of data points per processing chunk, at least 10. Default: 5000.
<code>overlap</code>	Number of overlapping points between consecutive chunks, less than chunk_size. NULL (default) computes chunk_size / 10, clamped to at least 1 and less than chunk_size.
<code>merge_strategy</code>	Strategy for reconciling overlapping chunk regions: "weighted_average" (default; alias: "weighted"), "average" (alias: "mean"), "take_first" (alias: "first"), or "take_last" (alias: "last").
<code>parallel</code>	Logical; enable parallel processing. Default: TRUE.
<code>outputs</code>	Streaming output choices: "se" (standard errors), "diagnostics", "residuals", "weights" (robustness weights), and/or "derivative". NULL (default) returns only the core result.
<code>intervals</code>	Interval options, created with <code>intervals_opts</code> (or a named list with any of confidence, prediction, bootstrap): e.g. <code>intervals = intervals_opts(confidence = 0.90, prediction = 0.99, bootstrap = 200)</code> . Confidence and prediction coverage levels are independent and may differ. NULL (default) disables intervals.
<code>seed</code>	Non-negative whole-number seed shared by cross-validation and bootstrap resampling for reproducible results. When cross-validation is unavailable, it controls bootstrap resampling only. NULL (default) uses a random seed.

Details

Best suited for datasets over 100,000 points, memory-constrained environments, or batch processing pipelines. For smaller datasets that fit in memory, see [Lowess](#); for point-by-point real-time data, see [OnlineLowess](#). When "diagnostics" is selected, residual_sd is the cumulative sample standard deviation of emitted residuals.

Confidence and prediction interval coverage levels are independent.

Overlapping regions between chunks are reconciled via merge_strategy:

Strategy	Alias	Behavior
"weighted_average" (default)	"weighted"	Distance-weighted blend
"average"	"mean"	Average overlapping values
"take_first"	"first"	Keep left chunk values
"take_last"	"last"	Keep right chunk values

Value

A StreamingLowess object.

Examples

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

Index

`add_point`, [2](#), [10](#)

`cv_opts`, [3](#), [9](#)

`finalize`, [4](#), [17](#)

`fit`, [4](#), [13](#)

`gpu_available`, [5](#), [6](#)

`install_gpu`, [5](#), [6](#)

`intervals_opts`, [7](#), [9](#), [11](#), [13](#), [18](#)

`Lowess`, [3](#), [7](#), [7](#), [12](#), [13](#), [19](#)

`OnlineLowess`, [7](#), [9](#), [10](#), [19](#)

`plot.LowessResult`, [12](#)

`predict.Lowess`, [7](#), [9](#), [13](#)

`print.Lowess`, [14](#)

`print.LowessResult`, [15](#)

`print.OnlineLowess`, [15](#)

`print.StreamingLowess`, [16](#)

`process_chunk`, [16](#), [17](#)

`StreamingLowess`, [7](#), [9](#), [12](#), [17](#)