Package: rfastloess (via r-universe)

July 12, 2026

Title High-Performance LOESS Smoothing for R

Version 0.9.0

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

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

Imports BiocGenerics

Suggests BiocStyle, 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

biocViews Preprocessing, StatisticalMethod, Regression, FunctionalPrediction, QualityControl, KEGG

Config/rextendr/version 0.4.2

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libclang-dev

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

Date/Publication 2026-07-12 22:41:11 UTC

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

RemoteRef HEAD

RemoteSha 8625a3236d39bbb0a2e84cb3d63d9214e6d8eabd

RemoteSubdir bindings/r

Contents

`rfastloess-package` *rfastloess: High-performance LOESS Smoothing for R*

Description

A high-performance LOESS (Locally Estimated Scatterplot Smoothing) implementation built on the Rust `fastLoess` crate.

For comprehensive documentation, see: https://github.com/thisisamirv/loess-project/tree/main/docs

Main Classes

- `Loess`: Primary interface for batch processing
- `StreamingLoess`: Chunked processing for large datasets
- `OnlineLoess`: Sliding window for real-time data

Documentation

For comprehensive documentation, tutorials, and API reference, see: https://loess.readthedocs.io/

Author(s)

Maintainer: Amir Valizadeh <thisisamirv@gmail.com> (ORCID) [funder]

Authors:

- Amir Valizadeh <thisisamirv@gmail.com> (ORCID) [funder]

See Also

Useful links:

- https://github.com/thisisamirv/loess-project
- Report bugs at https://github.com/thisisamirv/loess-project/issues

Examples

```
# Basic smoothing
x <- seq(1, 10, length.out = 100)
y <- sin(x) + rnorm(100, sd = 0.2)
model <- Loess(fraction = 0.3)
result <- model$fit(x, y)
plot(x, y)
lines(result$x, result$y, col = "red", lwd = 2)
```

Loess *LOESS Batch Smoothing*

Description

Create a stateful LOESS model for batch smoothing.

Usage

```
Loess(
    fraction = 0.67,
    iterations = 3L,
    weight_function = "tricube",
    robustness_method = "bisquare",
    scaling_method = "mad",
    boundary_policy = "extend",
    confidence_intervals = NULL,
    prediction_intervals = NULL,
    return_diagnostics = FALSE,
    return_residuals = FALSE,
    return_robustness_weights = FALSE,
    zero_weight_fallback = "use_local_mean",
    auto_converge = NULL,
    cv_fractions = NULL,
    cv_method = "kfold",
    cv_k = 5L,
    parallel = TRUE,
    degree = "linear",
    dimensions = 1L,
    distance_metric = "normalized",
    surface_mode = "interpolation",
    return_se = FALSE,
    weighted_metric_weights = NULL,
    cell = NULL,
    interpolation_vertices = NULL,
    boundary_degree_fallback = NULL,
    cv_seed = NULL
)
```

Arguments

fraction	Smoothing fraction (between 0 and 1).
iterations	Number of robustness iterations (non-negative integer). Default: 3.
weight_function	Kernel weight function. One of "tricube" (default), "gaussian", "uniform", "cosine", "epanechnikov", "biweight", "triangle".
robustness_method	Outlier downweighting method: "bisquare" (default), "huber", or "talwar".
scaling_method	Residual scale estimation for robustness weights: "mad" (default), "mar", or "mean".
boundary_policy	Boundary handling strategy: "extend" (default), "reflect", "zero", or "noboundary".
confidence_intervals	Confidence level for confidence intervals (e.g., 0.95). NULL (default) disables confidence intervals.
prediction_intervals	Confidence level for prediction intervals (e.g., 0.95). NULL (default) disables prediction intervals.
return_diagnostics	Logical; if TRUE, return fit-quality metrics (RMSE, MAE, R-squared, AIC, etc.). Default: FALSE.
return_residuals	Logical; if TRUE, return residuals in the result. Default: FALSE.
return_robustness_weights	Logical; if TRUE, return per-point robustness weights. Default: FALSE.
zero_weight_fallback	Fallback policy when all robustness weights drop to zero: "use_local_mean" (default), "return_original", or "return_none".
auto_converge	Convergence tolerance for early stopping of robustness iterations. NULL (default) disables early stopping.
cv_fractions	Numeric vector of candidate fractions for cross-validation. NULL (default) disables CV.
cv_method	Cross-validation method: "kfold" (default) or "loocv".
cv_k	Number of folds for k-fold CV. Default: 5.
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".

`return_se` Logical; if `TRUE`, compute hat-matrix statistics (effective degrees of freedom, leverage, standard errors). Default: `FALSE`.

`weighted_metric_weights` Numeric vector of per-dimension weights used when `distance_metric = "weighted"`. Length must equal `dimensions`. `NULL` (default) uses equal weights.

`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`, fall back to lower polynomial degree at boundaries when fitting at the requested degree fails. `NULL` (default) uses the library default.

`cv_seed` Integer seed for the cross-validation random number generator. `NULL` (default) uses a random seed.

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 <- model$fit(x, y)
plot(x, y)
lines(x, result$y, col = "red")
```

`Nullable` *Nullable Value Wrapper*

Description

Wraps a value to be passed to Rust as an Option.

Usage

```
Nullable(x)
```

Arguments

`x` Value to wrap or NULL.

Value

The value itself. This is a helper for rextendr conversion.

Examples

```
Nullable(5)
Nullable(NULL)
```

`OnlineLoess` *LOESS Online Smoothing*

Description

Create a stateful LOESS model for real-time online data.

Usage

```
OnlineLoess(
    fraction = 0.67,
    window_capacity = 1000L,
    min_points = 3L,
    iterations = 3L,
    weight_function = "tricube",
    robustness_method = "bisquare",
    scaling_method = "mad",
    boundary_policy = "extend",
    update_mode = "full",
    auto_converge = NULL,
    return_robustness_weights = FALSE,
    return_diagnostics = FALSE,
    return_residuals = FALSE,
    zero_weight_fallback = "use_local_mean",
    parallel = FALSE,
    degree = "linear",
    dimensions = 1L,
    distance_metric = "normalized",
    surface_mode = "interpolation",
    return_se = FALSE,
    confidence_intervals = NULL,
    prediction_intervals = NULL,
    weighted_metric_weights = NULL,
    cell = NULL,
    interpolation_vertices = NULL,
    boundary_degree_fallback = NULL
)
```

Arguments

`fraction`	Smoothing fraction (between 0 and 1).
`window_capacity`	Maximum number of points kept in the sliding window.

`min_points` Minimum number of points required before smoothing begins.

`iterations` Number of robustness iterations (non-negative integer). Default: 3.

`weight_function`
Kernel weight function. One of `"tricube"` (default), `"gaussian"`, `"uniform"`, `"cosine"`, `"epanechnikov"`, `"biweight"`, `"triangle"`.

`robustness_method`
Outlier downweighting method: `"bisquare"` (default), `"huber"`, or `"talwar"`.

`scaling_method` Residual scale estimation for robustness weights: `"mad"` (default), `"mar"`, or `"mean"`.

`boundary_policy`
Boundary handling strategy: `"extend"` (default), `"reflect"`, `"zero"`, or `"noboundary"`.

`update_mode` Window update strategy: `"full"` (default) re-smooths all window points after each addition, `"incremental"` updates only the newest point.

`auto_converge` Convergence tolerance for early stopping of robustness iterations. `NULL` (default) disables early stopping.

`return_robustness_weights`
Logical; if `TRUE`, return per-point robustness weights. Default: `FALSE`.

`return_diagnostics`
Logical; if `TRUE`, return fit-quality metrics (RMSE, MAE, R-squared, AIC, etc.). Default: `FALSE`.

`return_residuals`
Logical; if `TRUE`, return residuals in the result. Default: `FALSE`.

`zero_weight_fallback`
Fallback policy when all robustness weights drop to zero: `"use_local_mean"` (default), `"return_original"`, or `"return_none"`.

`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"`.

`return_se` Logical; if `TRUE`, compute hat-matrix statistics (effective degrees of freedom, leverage, standard errors). Default: `FALSE`.

`confidence_intervals`
Confidence level for confidence intervals (e.g., 0.95). `NULL` (default) disables confidence intervals.

`prediction_intervals`
Confidence level for prediction intervals (e.g., 0.95). `NULL` (default) disables prediction intervals.

`weighted_metric_weights`
Numeric vector of per-dimension weights used when `distance_metric = "weighted"`. Length must equal `dimensions`. `NULL` (default) uses equal weights.

`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`, fall back to lower polynomial degree at boundaries when fitting at the requested degree fails. `NULL` (default) uses the library default.

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)
for (i in seq_along(x)) {
    result <- model$add_point(x[i], y[i])
    if (!is.null(result)) cat("smoothed:", result$smoothed, "\n")
}
```

---

plot.LoessResult *Plot Loess Result*

Description

Plot Loess Result

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

The input object `x`, invisibly.

Examples

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

`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 <- model$fit(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)
```

StreamingLoess *LOESS Streaming Smoothing*

Description

Create a stateful LOESS model for streaming data.

Usage

```
StreamingLoess(
    fraction = 0.67,
    chunk_size = 5000L,
    overlap = NULL,
    iterations = 3L,
    weight_function = "tricube",
    robustness_method = "bisquare",
    scaling_method = "mad",
    boundary_policy = "extend",
    auto_converge = NULL,
    return_diagnostics = FALSE,
```

```
    return_residuals = FALSE,
    return_robustness_weights = FALSE,
    zero_weight_fallback = "use_local_mean",
    merge_strategy = "weighted_average",
    parallel = TRUE,
    degree = "linear",
    dimensions = 1L,
    distance_metric = "normalized",
    surface_mode = "interpolation",
    return_se = FALSE,
    confidence_intervals = NULL,
    prediction_intervals = NULL,
    weighted_metric_weights = NULL,
    cell = NULL,
    interpolation_vertices = NULL,
    boundary_degree_fallback = NULL
)
```

Arguments

`fraction`	Smoothing fraction (between 0 and 1).
`chunk_size`	Number of data points per processing chunk.
`overlap`	Number of overlapping points between consecutive chunks.
`iterations`	Number of robustness iterations (non-negative integer). Default: 3.
`weight_function`	Kernel weight function. One of `"tricube"` (default), `"gaussian"`, `"uniform"`, `"cosine"`, `"epanechnikov"`, `"biweight"`, `"triangle"`.
`robustness_method`	Outlier downweighting method: `"bisquare"` (default), `"huber"`, or `"talwar"`.
`scaling_method`	Residual scale estimation for robustness weights: `"mad"` (default), `"mar"`, or `"mean"`.
`boundary_policy`	Boundary handling strategy: `"extend"` (default), `"reflect"`, `"zero"`, or `"noboundary"`.
`auto_converge`	Convergence tolerance for early stopping of robustness iterations. `NULL` (default) disables early stopping.
`return_diagnostics`	Logical; if `TRUE`, return fit-quality metrics (RMSE, MAE, R-squared, AIC, etc.). Default: `FALSE`.
`return_residuals`	Logical; if `TRUE`, return residuals in the result. Default: `FALSE`.
`return_robustness_weights`	Logical; if `TRUE`, return per-point robustness weights. Default: `FALSE`.
`zero_weight_fallback`	Fallback policy when all robustness weights drop to zero: `"use_local_mean"` (default), `"return_original"`, or `"return_none"`.

`merge_strategy` Strategy for reconciling overlapping chunk regions: `"weighted_average"` (default), `"average"`, `"take_first"`, or `"take_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"`.

`return_se` Logical; if `TRUE`, compute hat-matrix statistics (effective degrees of freedom, leverage, standard errors). Default: `FALSE`.

`confidence_intervals`
Confidence level for confidence intervals (e.g., 0.95). `NULL` (default) disables confidence intervals.

`prediction_intervals`
Confidence level for prediction intervals (e.g., 0.95). `NULL` (default) disables prediction intervals.

`weighted_metric_weights`
Numeric vector of per-dimension weights used when `distance_metric = "weighted"`. Length must equal `dimensions`. `NULL` (default) uses equal weights.

`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`, fall back to lower polynomial degree at boundaries when fitting at the requested degree fails. `NULL` (default) uses the library default.

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 <- model$process_chunk(x[1:50], y[1:50])
res2 <- model$process_chunk(x[51:100], y[51:100])
final <- model$finalize()
```

Index