

Package: **weighteder** (via r-universe)

May 28, 2026

Type Package

Title Weighted Empirical Risk Minimization for Changepoint Regression

Version 0.1.0

Description R interface to the 'weighteder' package for 'Python', which provides 'scikit-learn'-style estimators for offline change point regression (data segmentation) via weighted empirical risk minimization. Supports least-squares, Huber, and logistic losses with fixed or cross-validated numbers of change points. Wraps 'Python' via 'reticulate'. Arpino and Venkataramanan (2026) <[doi:10.48550/arXiv.2604.11746](https://doi.org/10.48550/arXiv.2604.11746)>.

License Apache License (>= 2.0)

URL <https://github.com/gabrielarpino/weighteder-r>

BugReports <https://github.com/gabrielarpino/weighteder-r/issues>

Imports reticulate (>= 1.28)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements Python (>= 3.9), weighteder Python package

Config/pak/sysreqs libpng-dev python3

Repository <https://gabrielarpino.r-universe.dev>

Date/Publication 2026-04-21 02:04:15 UTC

RemoteUrl <https://github.com/gabrielarpino/weighteder-r>

RemoteRef HEAD

RemoteSha 1c90156fe5a03bb83b6ac78bb250a178cfd438cf

Contents

coef.werm_fit	2
predict.werm_fit	3
summary.werm_fit	3
weightederm_configure_python	4
werm_huber	4
werm_huber_cv	6
werm_least_squares	7
werm_least_squares_cv	9
werm_logistic	11
werm_logistic_cv	12

Index	14
--------------	-----------

coef.werm_fit	<i>Extract last-segment coefficients</i>
---------------	--

Description

Returns the coefficient vector of the unweighted base-loss refit on the last detected segment (the same coefficients used by predict()).

Usage

```
## S3 method for class 'werm_fit'
coef(object, ...)
```

Arguments

object	A werm_fit object.
...	Ignored.

Value

Named numeric vector of length p.

predict.werm_fit *Predict using the last-segment refit*

Description

Evaluates the unweighted base-loss model fitted on the last detected segment. For logistic estimators, returns class labels.

Usage

```
## S3 method for class 'werm_fit'
predict(object, newdata, type = "class", ...)
```

Arguments

object	A werm_fit object returned by any werm_* function.
newdata	Numeric matrix of shape (m, p).
type	Character. For logistic estimators: "class" (default) or "prob" (returns a (m, 2) probability matrix).
...	Ignored.

Value

Numeric vector of length m (regression) or character vector / probability matrix (logistic).

summary.werm_fit *Summarise a fitted WERM model*

Description

Summarise a fitted WERM model

Usage

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

Arguments

object	A werm_fit object.
...	Ignored.

Value

Invisibly returns object. Called for its printed side-effect.

weightederm_configure_python

Configure which Python environment weightederm uses

Description

A thin wrapper around `reticulate::use_python()` that must be called **before** any `werm_*` function if the default Python does not have `weightederm` installed.

Usage

```
weightederm_configure_python(python, required = TRUE)
```

Arguments

<code>python</code>	Path to the Python binary or virtual-environment directory.
<code>required</code>	Passed to <code>reticulate::use_python()</code> .

Value

Invisibly returns `NULL`. Called for side effects only.

Examples

```
if (nzchar(Sys.which("python"))) {
  weightederm_configure_python(Sys.which("python"), required = FALSE)
}
```

werm_huber

Fit a WERM changepoint model with Huber loss

Description

Like `werm_least_squares()` but uses the Huber loss, which is more robust to outliers than squared loss.

Usage

```
werm_huber(
  X,
  y,
  num_chgpts,
  delta = 1L,
  search_method = "efficient",
  fit_intercept = TRUE,
```

```

    epsilon = 1.35,
    max_iter = 100L,
    tol = 1e-05,
    penalty = "none",
    alpha = 0
  )

```

Arguments

<code>X</code>	Numeric matrix of shape (n, p) . Observations must be in order.
<code>y</code>	Numeric vector of length n . Ordered responses.
<code>num_chgpts</code>	Integer. Number of changepoints to detect.
<code>delta</code>	Integer. Minimum gap between candidate changepoints during search. Default 1. Rule of thumb: $\max(1L, \text{as.integer}(0.05 * \text{nrow}(X)))$.
<code>search_method</code>	Character. "efficient" (default) or "brute_force".
<code>fit_intercept</code>	Logical. Whether each segment model includes an intercept. Default TRUE.
<code>epsilon</code>	Numeric. Huber transition parameter. Default 1.35 (95 % Gaussian efficiency).
<code>max_iter</code>	Integer. Maximum L-BFGS-B iterations. Default 100.
<code>tol</code>	Numeric. Gradient-norm tolerance. Default $1e-5$.
<code>penalty</code>	Character. "none" (default), "l1", or "l2".
<code>alpha</code>	Numeric. Penalty strength. Default 0.

Value

An object of class `c("werm_huber", "werm_fit")`. Same elements as `werm_least_squares()`.

Examples

```

# Limit BLAS/OpenMP threads so example CPU time stays proportional to
# elapsed time on multicore CRAN machines.
Sys.setenv(
  OMP_NUM_THREADS = "1",
  OPENBLAS_NUM_THREADS = "1",
  MKL_NUM_THREADS = "1",
  BLAS_NUM_THREADS = "1"
)

if (nzchar(Sys.getenv("RETICULATE_PYTHON")) &&
    weightederm:::weightederm_examples_available("WERMHuber")) {
  set.seed(2)
  n <- 12L; p <- 1L; true_cp <- 6L
  X <- matrix(rnorm(n * p), n, p)
  y <- c(
    X[1:true_cp, , drop = FALSE] %%% 3,
    X[(true_cp + 1L):n, , drop = FALSE] %%% -3
  ) + rnorm(n, sd = 0.03)
  fit <- werm_huber(X, y, num_chgpts = 1L, delta = 1L,
    fit_intercept = FALSE, max_iter = 5L, tol = 1e-3)
}

```

```

    fit$changepoints
  }

```

werm_huber_cv

Fit a WERM changepoint model with Huber loss and CV selection

Description

Fit a WERM changepoint model with Huber loss and CV selection

Usage

```

werm_huber_cv(
  X,
  y,
  max_num_chgpts,
  delta = 1L,
  search_method = "efficient",
  cv = 5L,
  fit_intercept = TRUE,
  epsilon = 1.35,
  max_iter = 100L,
  tol = 1e-05,
  use_base_loss_for_cv = FALSE,
  penalty = "none",
  alpha = 0
)

```

Arguments

X	Numeric matrix of shape (n, p). Observations must be in order.
y	Numeric vector of length n. Ordered responses.
max_num_chgpts	Integer. Upper bound of the CV search grid.
delta	Integer. Minimum gap between candidate changepoints. Default 1.
search_method	Character. "efficient" (default) or "brute_force".
cv	Integer. Number of interleaved folds. Default 5.
fit_intercept	Logical. Default TRUE.
epsilon	Numeric. Huber transition parameter. Default 1.35.
max_iter	Integer. Maximum L-BFGS-B iterations. Default 100.
tol	Numeric. Gradient-norm tolerance. Default 1e-5.
use_base_loss_for_cv	Logical. If TRUE, uses squared loss for CV segment fits and scoring instead of the default absolute error. Default FALSE.
penalty	Character. Passed to the inner fixed model. Default "none".
alpha	Numeric. Penalty strength. Default 0.

Value

An object of class `c("werm_huber_cv", "werm_fit")`.

Examples

```
# Limit BLAS/OpenMP threads so example CPU time stays proportional to
# elapsed time on multicore CRAN machines.
Sys.setenv(
  OMP_NUM_THREADS = "1",
  OPENBLAS_NUM_THREADS = "1",
  MKL_NUM_THREADS = "1",
  BLAS_NUM_THREADS = "1"
)

if (nzchar(Sys.getenv("RETICULATE_PYTHON")) &&
    weightederm:::weightederm_examples_available("WERMHuberCV")) {
  set.seed(11)
  n <- 24L; p <- 2L
  X <- matrix(rnorm(n * p), n, p)
  y <- c(
    X[1:8, ] %*% c(4, 0),
    X[9:16, ] %*% c(-4, 0),
    X[17:24, ] %*% c(4, 0)
  ) + rnorm(n, sd = 0.02)
  fit <- werm_huber_cv(X, y, max_num_chgpts = 2L, cv = 3L, delta = 2L,
    fit_intercept = FALSE, max_iter = 20L)
  fit$best_num_chgpts
}
```

werm_least_squares *Fit a WERM changepoint model with squared loss*

Description

Detects `num_chgpts` changepoints in ordered regression data by minimising a Weighted Empirical Risk with squared loss.

Usage

```
werm_least_squares(
  X,
  y,
  num_chgpts,
  delta = 1L,
  search_method = "efficient",
  fit_intercept = TRUE,
  fit_solver = "direct",
  penalty = "none",
```

```

    alpha = 0
  )

```

Arguments

<code>X</code>	Numeric matrix of shape (n, p) . Observations must be in order.
<code>y</code>	Numeric vector of length n . Ordered responses.
<code>num_chgpts</code>	Integer. Number of changepoints to detect.
<code>delta</code>	Integer. Minimum gap between candidate changepoints during search. Default 1. Rule of thumb: $\max(1L, \text{as.integer}(0.05 * \text{nrow}(X)))$.
<code>search_method</code>	Character. "efficient" (default) or "brute_force".
<code>fit_intercept</code>	Logical. Whether each segment model includes an intercept. Default TRUE.
<code>fit_solver</code>	Character. "direct" (default, closed-form) or "lbfgsb" (gradient-based).
<code>penalty</code>	Character. "none" (default), "l1", or "l2".
<code>alpha</code>	Numeric. Penalty strength. Default 0.

Value

An object of class `c("werm_least_squares", "werm_fit")` with the following named elements:

- `changepoints` Integer vector of detected changepoints (**1-indexed**, R convention). Length equals `num_chgpts`.
- `num_chgpts` Integer. Number of detected changepoints.
- `num_signals` Integer. Number of segments (`num_chgpts + 1`).
- `objective` Numeric. Minimised WERM objective value.
- `last_segment_coef` Numeric vector. Coefficient of the unweighted base-loss refit on the last segment (used by `predict()`).
- `last_segment_intercept` Numeric or NULL.
- `signal_coefs` Matrix (`num_signals x p`). Stage-1 WERM coefficient estimates.
- `signal_intercepts` Numeric vector or NULL.
- `n_features_in` Integer. Number of features.

Examples

```

# Limit BLAS/OpenMP threads so example CPU time stays proportional to
# elapsed time on multicore CRAN machines.
Sys.setenv(
  OMP_NUM_THREADS = "1",
  OPENBLAS_NUM_THREADS = "1",
  MKL_NUM_THREADS = "1",
  BLAS_NUM_THREADS = "1"
)

if (nzchar(Sys.getenv("RETICULATE_PYTHON")) &&
    weightederm:::weightederm_examples_available("WERMLeastSquares")) {
  set.seed(1)

```

```

n <- 24L; p <- 2L; true_cp <- 12L
X <- matrix(rnorm(n * p), n, p)
y <- c(
  X[1:true_cp, ] %%% c(3, -1.5),
  X[(true_cp + 1L):n, ] %%% c(-3, 1.5)
) + rnorm(n, sd = 0.05)
fit <- werm_least_squares(X, y, num_chgpts = 1L, delta = 3L,
  fit_intercept = FALSE)

fit$changepts
}

```

werm_least_squares_cv *Fit a WERM changepoint model with squared loss and CV selection*

Description

Selects the number of changepoints from $\{0, \dots, \text{max_num_chgpts}\}$ by interleaved cross-validation, then refits on the full data.

Usage

```

werm_least_squares_cv(
  X,
  y,
  max_num_chgpts,
  delta = 1L,
  search_method = "efficient",
  cv = 5L,
  fit_intercept = TRUE,
  use_base_loss_for_cv = FALSE,
  penalty = "none",
  alpha = 0
)

```

Arguments

X	Numeric matrix of shape (n, p). Observations must be in order.
y	Numeric vector of length n. Ordered responses.
max_num_chgpts	Integer. Upper bound of the CV search grid.
delta	Integer. Minimum gap between candidate changepoints. Default 1.
search_method	Character. "efficient" (default) or "brute_force".
cv	Integer. Number of interleaved folds. Default 5.
fit_intercept	Logical. Default TRUE.
use_base_loss_for_cv	Logical. If TRUE, uses squared loss for CV segment fits and scoring instead of the default absolute error. Default FALSE.

penalty Character. Passed to the inner fixed model. Default "none".
 alpha Numeric. Penalty strength. Default 0.

Value

An object of class `c("werm_least_squares_cv", "werm_fit")` with all elements from `werm_least_squares()` plus:

`best_num_chgpts` Integer. CV-selected number of changepoints.
`best_score` Numeric. Mean held-out score for the best model.
`cv_results` Data frame with columns `num_chgpts` and `mean_test_score`.
`num_chgpts_grid` Integer vector. Full CV search grid.
`segment_bounds` List of integer pairs `[start, stop]` (1-indexed, half-open).
`segment_coefs` Matrix (`num_signals` x `p`).
`segment_intercepts` Numeric vector or NULL.

Examples

```
# Limit BLAS/OpenMP threads so example CPU time stays proportional to
# elapsed time on multicore CRAN machines.
Sys.setenv(
  OMP_NUM_THREADS = "1",
  OPENBLAS_NUM_THREADS = "1",
  MKL_NUM_THREADS = "1",
  BLAS_NUM_THREADS = "1"
)

if (nzchar(Sys.getenv("RETICULATE_PYTHON")) &&
    weightederm:::weightederm_examples_available("WERMLEastSquaresCV")) {
  set.seed(10)
  n <- 30L; p <- 2L
  X <- matrix(rnorm(n * p), n, p)
  y <- c(
    X[1:10, ] %*% c(3.5, 0),
    X[11:20, ] %*% c(-3.5, 0),
    X[21:30, ] %*% c(3.5, 0)
  ) + rnorm(n, sd = 0.05)
  fit <- werm_least_squares_cv(X, y, max_num_chgpts = 2L, cv = 3L,
                              delta = 3L, fit_intercept = FALSE)

  fit$best_num_chgpts
  fit$changepoints
  fit$cv_results
}
```

werm_logistic	<i>Fit a WERM changepoint model with binary logistic loss</i>
---------------	---

Description

Detects num_chgpts changepoints in ordered binary classification data.

Usage

```
werm_logistic(
  X,
  y,
  num_chgpts,
  delta = 1L,
  search_method = "efficient",
  fit_intercept = TRUE,
  max_iter = 100L,
  tol = 1e-05,
  penalty = "l2",
  alpha = 1
)
```

Arguments

X	Numeric matrix of shape (n, p). Observations must be in order.
y	Integer or factor vector of binary labels (two unique values).
num_chgpts	Integer. Number of changepoints to detect.
delta	Integer. Minimum gap between candidate changepoints during search. Default 1. Rule of thumb: $\max(1L, \text{as.integer}(0.05 * \text{nrow}(X)))$.
search_method	Character. "efficient" (default) or "brute_force".
fit_intercept	Logical. Whether each segment model includes an intercept. Default TRUE.
max_iter	Integer. Maximum L-BFGS-B iterations. Default 100.
tol	Numeric. Gradient-norm tolerance. Default 1e-5.
penalty	Character. Default "l2" (prevents divergence on separable segments). Set "none" only when segments are not separable.
alpha	Numeric. Default 1.0.

Value

An object of class `c("werm_logistic", "werm_fit")`. Contains all elements from `werm_least_squares()` plus:

`classes` Character vector of length 2. The two class labels in sorted order. `classes[2]` is the positive class.

Examples

```

# Limit BLAS/OpenMP threads so example CPU time stays proportional to
# elapsed time on multicore CRAN machines.
Sys.setenv(
  OMP_NUM_THREADS = "1",
  OPENBLAS_NUM_THREADS = "1",
  MKL_NUM_THREADS = "1",
  BLAS_NUM_THREADS = "1"
)

if (nzchar(Sys.getenv("RETICULATE_PYTHON")) &&
    weightederm:::weightederm_examples_available("WERMLogistic")) {
  set.seed(3)
  n <- 30L; p <- 2L; true_cp <- 15L
  X <- matrix(rnorm(n * p), n, p)
  eta <- c(
    X[1:true_cp, ] %*% c(3, -3),
    X[(true_cp + 1L):n, ] %*% c(-3, 3)
  )
  y <- rbinom(n, 1L, 1 / (1 + exp(-eta)))
  fit <- werm_logistic(X, y, num_chgpts = 1L, delta = 3L,
    fit_intercept = FALSE, max_iter = 100L)

  fit$changepoints
  fit$classes
}

```

werm_logistic_cv

Fit a WERM changepoint model with logistic loss and CV selection

Description

Fit a WERM changepoint model with logistic loss and CV selection

Usage

```

werm_logistic_cv(
  X,
  y,
  max_num_chgpts,
  delta = 1L,
  search_method = "efficient",
  cv = 5L,
  fit_intercept = TRUE,
  max_iter = 100L,
  tol = 1e-05,
  use_base_loss_for_cv = FALSE,
  penalty = "l2",
  alpha = 1
)

```

Arguments

X	Numeric matrix of shape (n, p). Observations must be in order.
y	Integer or factor vector of binary labels.
max_num_chgpts	Integer. Upper bound of the CV search grid.
delta	Integer. Minimum gap between candidate changepoints. Default 1.
search_method	Character. "efficient" (default) or "brute_force".
cv	Integer. Number of interleaved folds. Default 5.
fit_intercept	Logical. Default TRUE.
max_iter	Integer. Maximum L-BFGS-B iterations. Default 100.
tol	Numeric. Gradient-norm tolerance. Default 1e-5.
use_base_loss_for_cv	Logical. If TRUE, uses logistic loss for CV segment scoring instead of the default absolute error. Default FALSE.
penalty	Character. Default "l2".
alpha	Numeric. Default 1.0.

Value

An object of class c("worm_logistic_cv", "worm_fit"). Contains all CV elements plus classes (character vector of length 2).

Examples

```
# Limit BLAS/OpenMP threads so example CPU time stays proportional to
# elapsed time on multicore CRAN machines.
Sys.setenv(
  OMP_NUM_THREADS = "1",
  OPENBLAS_NUM_THREADS = "1",
  MKL_NUM_THREADS = "1",
  BLAS_NUM_THREADS = "1"
)

if (nzchar(Sys.getenv("RETICULATE_PYTHON")) &&
    weightederm:::weightederm_examples_available("WERMLogisticCV")) {
  set.seed(12)
  n <- 30L; p <- 2L
  X <- matrix(rnorm(n * p), n, p)
  eta <- c(
    X[1:10, ] %*% c(3, -3),
    X[11:20, ] %*% c(-3, 3),
    X[21:30, ] %*% c(3, -3)
  )
  y <- rbinom(n, 1L, 1 / (1 + exp(-eta)))
  fit <- worm_logistic_cv(X, y, max_num_chgpts = 2L, cv = 3L, delta = 3L,
    fit_intercept = FALSE, max_iter = 100L)

  fit$best_num_chgpts
  fit$changepoints
}
```

Index

`coef.werm_fit`, [2](#)

`predict.werm_fit`, [3](#)

`reticulate::use_python()`, [4](#)

`summary.werm_fit`, [3](#)

`weightederm_configure_python`, [4](#)

`werm_huber`, [4](#)

`werm_huber_cv`, [6](#)

`werm_least_squares`, [7](#)

`werm_least_squares()`, [4](#), [5](#), [10](#), [11](#)

`werm_least_squares_cv`, [9](#)

`werm_logistic`, [11](#)

`werm_logistic_cv`, [12](#)