Package: jumps (via r-universe)

October 30, 2026

Type Package

Title Hodrick-Prescott Filter and Spline Smoothing with Jumps

Version 2.0.0

Date 2026-10-27

Maintainer Matteo Pelagatti <matteo.pelagatti@unimib.it>

Description A set of functions to compute the Hodrick-Prescott (HP) filter and spline smoothing with automatically selected jumps. The original HP filter and spline smoothing extract a smooth signal, and our version allows for a small number of automatically identified jumps. Smoothing splines with discontinuities are obtained either by time warping, or by the dynamic-programming algorithm of Storath and Weinmann (2024) <doi:10.1080/10618600.2023.2262000>. See Pelagatti, Maranzano and Gatto Ronchieri (2026) <doi:10.48550/arXiv.2610.33137> for details on the smoothing splines with discontinuities, and Maranzano and Pelagatti (2025) <doi:10.1016/j.econmod.2025.107132> for details on the HP filter with jumps.

License GPL-3

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.10), stats, graphics, grDevices, nloptr, utils

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ggplot2, xts

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

LazyData true

URL https://github.com/matteopelagatti/jumps

BugReports https://github.com/matteopelagatti/jumps/issues

NeedsCompilation yes

Author Matteo Pelagatti [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1860-7535>), Paolo Maranzano [aut, cph] (ORCID: <https://orcid.org/0000-0002-9228-2759>), Giulio Cesare Gatto Ronchieri [ctb]

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

Date/Publication 2026-10-30 14:24:22 UTC

RemoteUrl https://github.com/cran/jumps

RemoteRef HEAD

RemoteSha 716ebfc388243adb06cd399dcfbfef9b17d1f658

Contents

auto_hpfj . . . 3
auto_hpfj_fix . . . 4
auto_hpfjx . . . 5
auto_ssj_atw . . . 6
auto_ssj_atw_cv . . . 8
auto_ssj_atw_gcv . . . 10
auto_ssj_cssd . . . 12
auto_ssj_fix . . . 14
auto_ssj_kfs . . . 15
auto_ssj_mle . . . 17
BIC.hpj . . . 18
demand_temp . . . 19
dummyseas . . . 20
employed_IT . . . 21
hpfj . . . 21
hpfj_fix . . . 23
hpfjx . . . 25
hpj . . . 26
llt . . . 28
load_temp . . . 30
logLik.hpj . . . 31
mse . . . 31
nobs.hpj . . . 32
plot.hpj . . . 33
plot.ssj . . . 34
predict.ssj . . . 36
predict_cssd_dp . . . 37
print.hpj . . . 38
print.ssj . . . 39
ssj_atw . . . 40
ssj_breaks . . . 43
ssj_cssd . . . 44
ssj_cssd_dp . . . 46

ssj_fix 47
ssj_kfs 50
ssj_mle 52
trigseas 56

Index **57**

`auto_hpfj` *Automatic selection of the optimal HP filter with jumps*

Description

The regularization constant for the HP filter with jumps is the maximal sum of standard deviations of the level disturbance. This value has to be passed to the `hpfj` function. `auto_hpfj` runs `hpfj` on a grid of regularization constants and returns the output of `hpfj` selected by the chosen information criterion.

Usage

```
auto_hpfj(
  y,
  grid = seq(0, sd(y, na.rm = TRUE) * 10, sd(y, na.rm = TRUE)/10),
  ic = c("bic", "hq", "aic", "aicc"),
  edf = TRUE
)
```

Arguments

`y`	numeric vector containing the time series (missing values allowed).
`grid`	numeric vector containing the grid for the argument `maxsum` of the `hpfj` function.
`ic`	string with the information criterion used for the choice: the default is `"bic"` (simulations show this is the best choice), but `"hq"`, `"aic"` and `"aicc"` are also available.
`edf`	logical: `TRUE` (default) if the number of degrees of freedom should be computed as "effective degrees of freedom" (Efron, 1986), as opposed to a more traditional way (although not supported by theory) when `FALSE`.

Value

The output of the `hpfj` function corresponding to the best choice according to the selected information criterion.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

See Also

hpfj, hpj.

Examples

```
mod <- auto_hpfj(Nile)
plot(as.numeric(Nile), type = "l")
lines(mod$smoothed_level, col = "red")
```

`auto_hpfj_fix` *Automatic selection of the optimal HP filter with jumps and fixed smoothing constant*

Description

The regularization constant for the HP filter with jumps is the maximal sum of standard deviations of the level disturbance. This value has to be passed to the `hpfj_fix` function. `auto_hpfj_fix` runs `hpfj_fix` on a grid of regularization constants and returns the output of `hpfj_fix` selected by the chosen information criterion.

Usage

```
auto_hpfj_fix(
  y,
  lambda,
  grid = seq(0, sd(y, na.rm = TRUE) * 10, sd(y, na.rm = TRUE)/10),
  ic = c("bic", "hq", "aic", "aicc"),
  edf = TRUE
)
```

Arguments

`y`	numeric vector containing the time series (missing values allowed).
`lambda`	smoothing constant; see `hpfj_fix` for the accepted values.
`grid`	numeric vector containing the grid for the argument `maxsum` of the `hpfj_fix` function.
`ic`	string with the information criterion used for the choice: the default is `"bic"` (simulations show this is the best choice), but `"hq"`, `"aic"` and `"aicc"` are also available.
`edf`	logical: `TRUE` (default) if the number of degrees of freedom should be computed as "effective degrees of freedom" (Efron, 1986), as opposed to a more traditional way (although not supported by theory) when `FALSE`.

Value

The output of the hpfj_fix function corresponding to the best choice according to the selected information criterion.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

See Also

hpfj_fix, hpj.

Examples

```
mod <- auto_hpfj_fix(Nile, "annual")
plot(as.numeric(Nile), type = "l")
lines(mod$smoothed_level, col = "red")
```

---

auto_hpfjx *Automatic selection of the optimal HP filter with jumps and regressors*

Description

The regularization constant for the HP filter with jumps is the maximal sum of standard deviations of the level disturbance. This value has to be passed to the hpfjx function. auto_hpfjx runs hpfjx on a grid of regularization constants and returns the output of hpfjx selected by the chosen information criterion.

Usage

```
auto_hpfjx(
  y,
  X,
  grid = seq(0, sd(y, na.rm = TRUE) * 10, sd(y, na.rm = TRUE)/10),
  ic = c("bic", "hq", "aic", "aicc"),
  edf = TRUE
)
```

Arguments

y	numeric vector containing the time series (missing values allowed).
X	numeric matrix with the regressors in the columns.
grid	numeric vector containing the grid for the argument maxsum of the hpfjx function.

ic	string with the information criterion used for the choice: the default is "bic" (simulations show this is the best choice), but "hq", "aic" and "aicc" are also available.
edf	logical: TRUE (default) if the number of degrees of freedom should be computed as "effective degrees of freedom" (Efron, 1986), as opposed to a more traditional way (although not supported by theory) when FALSE.

Details

This function is still experimental and needs more testing, since it does not always work as expected. For this reason the wrapper hpj does not allow regressors at the moment.

Value

The output of the hpfjx function corresponding to the best choice according to the selected information criterion.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

See Also

hpfjx, hpfj.

Examples

```
y <- log(AirPassengers)
n <- length(y)
mod <- auto_hpfjx(y, trigseas(n, 12))
trend <- ts(mod$smoothed_level, start(y), frequency = 12)
plot(y)
lines(trend, col = "red")
```

---

auto_ssj_atw — *Automatic selection of the optimal smoothing spline with jumps using the alternating time-warping algorithm*

Description

The regularization parameter of the ATW algorithm is the maximal sum of additional times. This value has to be passed to the ssj_atw function. auto_ssj_atw runs ssj_atw on a grid of regularization constants and returns the output of ssj_atw selected by the chosen information criterion. Each grid point is warm-started from the solution at the previous point, and the search stops early when the selected criterion increases for three consecutive grid values.

Usage

```
auto_ssj_atw(
  y,
  x,
  lambda,
  weights = NULL,
  grid = seq(0, max(x) - min(x), length.out = 21L),
  ic = c("ebic", "bic", "hq", "aic", "aicc"),
  max_iter = 100,
  tol = 1e-04,
  learning_rate = 0.1,
  ebic_xi = 0.25
)
```

Arguments

y	numeric vector with the response (no missing values allowed).
x	numeric vector with the covariate (no missing values allowed).
lambda	numeric scalar with the smoothing parameter.
weights	optional numeric vector of positive observation weights, passed to ssj_atw; see its help page.
grid	numeric vector of maxsum values (in the units of x) to search; the default is 21 equally spaced values from 0 to the range of x.
ic	string with the information criterion used for the choice: the default is "ebic" (recommended when jumps are expected to be rare relative to the n-1 candidate locations, see ebic_xi), but "bic", "hq", "aic" and "aicc" are also available.
max_iter	maximum number of iterations of ssj_atw (default 100).
tol	tolerance for the convergence of ssj_atw (default 1e-4).
learning_rate	learning rate of ssj_atw (default 0.1).
ebic_xi	numeric scalar in [0,1] passed to ssj_atw: strength of the extended-BIC search penalty (default 0.25; only relevant when ic = "ebic"). See the "Choosing ebic_xi" section of ssj_atw for how it was calibrated, why it differs from the likelihood-based estimators, and for the rule that adapts it to the sample size.

Value

The output of the ssj_atw function corresponding to the best choice according to the selected information criterion.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

ssj_atw.

Examples

```
plot(faithful$eruptions, faithful$waiting)
of_jum <- auto_ssj_atw(y = faithful$waiting,
                       x = faithful$eruptions,
                       lambda = 1)
lines(x = of_jum$x, y = of_jum$smoothed_level, col = "red", lwd = 2)
```

`auto_ssj_atw_cv` *Smoothing splines with jumps by alternating time warping, with the smoothing constant chosen by cross-validation*

Description

Selects `lambda` by K-fold cross-validation over the smoothing constant and the warping budget jointly, keeps *only* the selected `lambda`, and passes it to `auto_ssj_atw`, which chooses the budget M by information criterion. Like `auto_ssj_atw_gcv` it makes the alternating algorithm self-contained; it is markedly more expensive but does not need the floor on the grid that generalized cross-validation requires.

Usage

```
auto_ssj_atw_cv(
  y,
  x,
  weights = NULL,
  lambda_grid = NULL,
  nfolds = 5L,
  folds = NULL,
  grid = seq(0, max(x) - min(x), length.out = 21L),
  ic = c("ebic", "bic", "hq", "aic", "aicc"),
  max_iter = 100,
  tol = 1e-04,
  learning_rate = 0.1,
  ebic_xi = 0.25
)
```

Arguments

`y`	numeric vector with the response (no missing values allowed).
`x`	numeric vector with the covariate (no missing values allowed).
`weights`	optional numeric vector of positive observation weights, passed to `ssj_atw` and used to weight the held-out squared errors; see its help page.
`lambda_grid`	numeric vector of candidate smoothing constants, in the units of `x`; the default is as in `auto_ssj_atw_gcv`. No floor is needed here, so it may be extended downwards freely.

`nfolds` integer number of cross-validation folds (default 5). Folds are drawn over the *distinct* values of `x`, so all replicates at a tied covariate value are held out together; `nfolds` must therefore not exceed the number of distinct values. See Details.

`folds` optional integer vector of fold labels, one per observation, for reproducibility or for grouped designs; overrides `nfolds`. A partition that puts replicates of the same `x` in different folds draws a warning, since the resulting criterion is optimistic.

`grid` numeric vector of `maxsum` values, used both for the cross-validation surface and for the final selection of M.

`ic` string with the information criterion used to choose M, passed to `auto_ssj_atw` (default `"ebic"`).

`max_iter`, `tol`, `learning_rate`, `ebic_xi` passed to `ssj_atw`.

Details

Why the budget is re-selected afterwards. The cross-validated budget is not used. Cross-validation estimates prediction error, and prediction error is nearly indifferent to a handful of spurious warps: on the calibration design, selecting M this way declared substantially more discontinuities than the information criterion at the same accuracy. The smoothing constant is a different matter – it governs a global bias-variance trade-off to which every held-out point contributes – and there cross-validation is competitive with maximum likelihood. Hence the split: cross-validate `lambda`, leave M to the criterion.

Tied values of `x` require a choice about the partition. Folds are drawn over the distinct values of `x`, so every replicate at a tied covariate value is held out together. Drawing them over the observations instead would leave replicates of a held-out site in the training sample, where `ssj_atw` collapses them and pins the fit at that very covariate value; the criterion would then estimate the error of predicting a fresh observation at an already observed `x` rather than at a new one, and is optimistic for the present purpose – by roughly a third under heavy replication. For a design with no ties the two partitions coincide.

Value

The object returned by `auto_ssj_atw` at the selected `lambda`, of class `ssj`, with three extra slots:

- `lambda_path`: data frame of the candidate `lambda` values and their best cross-validated mean squared prediction error.
- `cv_surface`: the full `lambda` by `maxsum` matrix of cross-validated errors.
- `lambda_source`: the string `"cv"`.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

`auto_ssj_atw_gcv`, `auto_ssj_atw`, `ssj_atw`.

Examples

```
set.seed(11)
x <- sort(runif(60))
y <- sin(2*pi*x) + 0.6*(x > 0.45) + rnorm(60, sd = 0.1)
# a short grid keeps the example quick
fit <- auto_ssj_atw_cv(y, x, nfolds = 3,
                       lambda_grid = 10^seq(-6, -3, length.out = 4),
                       grid = seq(0, 1, length.out = 6))
plot(x, y, pch = 16, cex = 0.6, col = "grey60")
lines(fit$x, fit$smoothed_level, col = "steelblue", lwd = 2)
```

`auto_ssj_atw_gcv` *Smoothing splines with jumps by alternating time warping, with the smoothing constant chosen by generalized cross-validation*

Description

Selects `lambda` by minimising the generalized cross-validation criterion of a plain smoothing spline – that is, of `ssj_atw` with the warping budget held at zero – and then passes it to `auto_ssj_atw`, which chooses the budget M by information criterion in the usual way. Unlike the two-step recipe in which `lambda` comes from `ssj_mle`, nothing here depends on the likelihood-based estimator, so the alternating algorithm stands on its own.

Usage

```
auto_ssj_atw_gcv(
  y,
  x,
  weights = NULL,
  lambda_grid = NULL,
  grid = seq(0, max(x) - min(x), length.out = 21L),
  ic = c("ebic", "bic", "hq", "aic", "aicc"),
  max_iter = 100,
  tol = 1e-04,
  learning_rate = 0.1,
  ebic_xi = 0.25
)
```

Arguments

`y`	numeric vector with the response (no missing values allowed).
`x`	numeric vector with the covariate (no missing values allowed).
`weights`	optional numeric vector of positive observation weights, passed to `ssj_atw`; see its help page.

`lambda_grid` numeric vector of candidate smoothing constants, in the units of `x`. The default is nine values spanning `1e-6` to `1e-2` once `x` has been scaled to unit range, which is the range over which the criterion behaves; values below the floor make it degenerate. A warning is issued when the selected value is an endpoint of the grid.

`grid` numeric vector of `maxsum` values passed to `auto_ssj_atw`; the default is 21 equally spaced values from 0 to the range of `x`.

`ic` string with the information criterion used to choose M, passed to `auto_ssj_atw` (default `"ebic"`).

`max_iter`, `tol`, `learning_rate`, `ebic_xi` passed to `ssj_atw`.

Details

The cost is one fit per candidate `lambda`, because `ssj_atw` already returns the exact effective degrees of freedom, the trace of the smoothing matrix, so the criterion is available in closed form and no resampling is needed. That makes this route roughly two orders of magnitude cheaper than `auto_ssj_atw_cv`.

Why the budget is held at zero. Minimising the criterion over `lambda` and M jointly does not work. The warps give the fit a way to interpolate: the effective degrees of freedom approach the number of observations, the denominator of the criterion collapses, and the smallest `lambda` on offer wins. On the calibration design that cost 58% in root mean squared error and reduced the Matthews correlation of the detected discontinuities from 0.54 to 0.08. With no budget there are no warps to exploit and the classical behaviour is restored.

Know what this routine can and cannot do. At zero budget the spline has no way to represent a discontinuity except by bending sharply, so when the data contain clear jumps and little noise the criterion keeps improving as `lambda` falls and the selection lands on the floor of the grid. On a design with three jumps this happened in 83% of samples at $n = 100$ and in every sample at $n = 200$ when $\sigma = 0.05$, against 7% and 23% at $\sigma = 0.15$ and about 1% with no jumps at all. In that regime the routine is not choosing `lambda`; it is returning the floor, and the fit is only as good as that floor happens to be. The resulting accuracy is in fact competitive – root mean squared error within a few percent of the alternatives – but the value is not a minimum of the criterion, and the warning issued in that case should be taken at face value.

`auto_ssj_atw_cv` does not have this failure mode: its floor bound in 0 to 23% of the same samples and the `lambda` it chooses tracks the maximum-likelihood value closely. Prefer it when the cost is affordable, and prefer this routine when it is not, or when the signal is noisy enough that the degeneracy does not bite.

Value

The object returned by `auto_ssj_atw` at the selected `lambda`, of class `ssj`, with two extra slots:

- `lambda_path`: data frame of the candidate `lambda` values and their generalized cross-validation criterion.
- `lambda_source`: the string `"gcv"`.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

Craven, P. and Wahba, G. (1978). Smoothing noisy data with spline functions. *Numerische Mathematik*, **31**(4), 377–403. doi:10.1007/BF01404567

See Also

auto_ssj_atw_cv, auto_ssj_atw, ssj_atw.

Examples

```
set.seed(11)
x <- sort(runif(80))
y <- sin(2*pi*x) + 0.6*(x > 0.45) + rnorm(80, sd = 0.1)
fit <- auto_ssj_atw_gcv(y, x)
plot(x, y, pch = 16, cex = 0.6, col = "grey60")
lines(fit$x, fit$smoothed_level, col = "firebrick", lwd = 2)
fit$pars["lambda"]
```

---

auto_ssj_cssd *Automatic CSSD with K-fold cross-validation of the parameters*

Description

Selects the stiffness parameter p and the jump-count penalty γ jointly by K-fold cross-validation, as in Section 3.4 of Storath and Weinmann (2024). A coarse two-dimensional grid search over (p, γ) is followed by a Nelder-Mead refinement on the reparameterised scale $(\mathrm{logit}(p), \mathrm{logit}(q))$, with $\gamma = p\,q/(1-q)$.

Usage

```
auto_ssj_cssd(
  y,
  x = NULL,
  k_folds = 5L,
  delta = 1,
  p_init = 0.99,
  gamma_init = 1,
  gamma_grid = NULL,
  maxit_nm = 200L
)
```

Arguments

y	numeric vector of observations (no missing values).
x	numeric vector of sampling locations (no missing values). If NULL (default), regular spacing 1:n is assumed.
k_folds	number of cross-validation folds (default 5, as in the paper).
delta	positive scalar or vector of length n with the noise standard deviations (default 1).
p_init	initial stiffness (default 0.99, as in the paper).
gamma_init	initial jump penalty, in the units of y (default 1). The search itself is conducted on y standardised to unit standard deviation, so that the range of jump penalties explored does not depend on the units of the response; gamma_init is standardised with it, and the gamma reported in cv_par is back in the units of y. A selected penalty at the edge of the search range draws a warning, since the number of discontinuities is then set by the range rather than by the data.
gamma_grid	optional vector of candidate jump penalties for the coarse search, in units of var(y); the default spans 1e-3 to 1e3 over 25 logarithmically spaced points, which covers the range of noise-to-signal ratios met in practice. The Nelder-Mead refinement is not confined to it, but a coarse grid that misses the optimum by orders of magnitude can leave the refinement in a local dip.
maxit_nm	maximum number of Nelder-Mead iterations (default 200).

Details

The fold assignment is drawn with sample, so the result depends on the state of the random number generator: call set.seed beforehand for reproducible output. Note also that the underlying solver is $O(n^2)$ in both time and memory, and that the cross-validation calls it k_folds times per grid point.

Value

The output of ssj_cssd_dp at the cross-validated optimal (p, γ), with two extra slots:

- cv_score: the minimised cross-validation score.
- cv_par: named vector with the selected p and gamma.

References

Storath, M. and Weinmann, A. (2024). Smoothing splines for discontinuous signals. *Journal of Computational and Graphical Statistics*, **33**(2), 651–664. doi:10.1080/10618600.2023.2262000

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

ssj_cssd_dp, predict_cssd_dp.

Examples

```
set.seed(1)
x   <- sort(runif(60))
y   <- sin(4 * pi * x) + c(rep(0, 30), rep(1, 30)) + rnorm(60, sd = 0.1)
fit <- auto_ssj_cssd(y, x)
fit$cv_par
plot(x, y)
lines(fit$x, fit$smoothed_level, col = "darkgreen", lwd = 2)
```

`auto_ssj_fix` *Automatic selection of the optimal smoothing spline with jumps and fixed smoothing parameter*

Description

The regularization parameter is the maximal sum of standard deviations added to the static standard deviation. This value has to be passed to the `ssj_fix` function. `auto_ssj_fix` runs `ssj_fix` on a grid of regularization constants and returns the output of `ssj_fix` selected by the chosen information criterion.

Usage

```
auto_ssj_fix(
  y,
  x,
  lambda,
  weights = NULL,
  grid = seq(0, sd(y, na.rm = TRUE)/mean(diff(x)) * 10, sd(y, na.rm =
    TRUE)/mean(diff(x))/10),
  ic = c("bic", "hq", "aic", "aicc"),
  edf = TRUE,
  last_delta = 1
)
```

Arguments

`y`	numeric vector with the response (missing values allowed).
`x`	numeric vector with the covariate (no missing values allowed).
`lambda`	numeric scalar with the smoothing parameter.
`weights`	optional numeric vector of positive observation weights, passed to the fitter; see its help page.
`grid`	numeric vector containing the grid for the argument `maxsum` of the `ssj_fix` function.

`ic`	string with the information criterion used for the choice: the default is `"bic"` (simulations show this is the best choice), but `"hq"`, `"aic"` and `"aicc"` are also available.
`edf`	logical: `TRUE` (default) if the number of degrees of freedom should be computed as "effective degrees of freedom" (Efron, 1986), as opposed to a more traditional way (although not supported by theory) when `FALSE`.
`last_delta`	numeric scalar with the difference $x_{n+1} - x_n$; not relevant for most applications and defaults to 1.

Value

The output of the `ssj_fix` function corresponding to the best choice according to the selected information criterion.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

`ssj_fix`.

Examples

```
# a short, explicit grid keeps the example fast; the default grid has 101
# points and is correspondingly slower
plot(faithful$eruptions, faithful$waiting)
of_jum <- auto_ssj_fix(y = faithful$waiting,
                       x = faithful$eruptions,
                       lambda = 1,
                       grid = seq(0, 400, length.out = 5))
lines(x = of_jum$x, y = of_jum$smoothed_level, col = "red", lwd = 2)
```

---

auto_ssj_kfs *Automatic selection of the optimal time-warping smoothing spline with jumps using the Kalman filter and smoother*

Description

Runs `ssj_kfs` over a grid of `maxsum` values and returns the solution with the lowest value of the selected information criterion. Each grid point is warm-started from the previous optimum, and the search stops early when the criterion increases for three consecutive grid values.

Usage

```
auto_ssj_kfs(
  y,
  x,
  lambda,
  weights = NULL,
  grid = seq(0, max(x) - min(x), length.out = 21L),
  ic = c("ebic", "bic", "hq", "aic", "aicc"),
  edf = TRUE,
  last_delta = 1,
  ebic_xi = 0.15
)
```

Arguments

y	numeric vector with the response (missing values allowed).
x	numeric vector with the covariate (no missing values allowed).
lambda	positive scalar with the smoothing parameter.
weights	optional numeric vector of positive observation weights, passed to the fitter; see its help page.
grid	numeric vector of maxsum values to search; the default is 21 equally spaced values from 0 to the range of x.
ic	string with the information criterion used for the choice: the default is "ebic" (recommended when jumps are expected to be rare relative to the n-1 candidate locations, see ebic_xi), but "bic", "hq", "aic" and "aicc" are also available.
edf	logical: passed to ssj_kfs (default TRUE).
last_delta	numeric scalar: passed to ssj_kfs (default 1).
ebic_xi	numeric scalar in [0,1] passed to ssj_kfs: strength of the extended-BIC search penalty (default 0.15; only relevant when ic = "ebic"). See the "Choosing ebic_xi" section of ssj_kfs for how it was calibrated, and why it differs from the value used by auto_ssj_mle.

Value

The output of the ssj_kfs function corresponding to the best choice according to the selected information criterion.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

ssj_kfs.

Examples

```
plot(faithful$eruptions, faithful$waiting)
of_kfs <- auto_ssj_kfs(y = faithful$waiting,
                       x = faithful$eruptions,
                       lambda = 1)
lines(x = of_kfs$x, y = of_kfs$smoothed_level, col = "blue", lwd = 2)
```

`auto_ssj_mle` *Automatic selection of the optimal time-warping smoothing spline with jumps using maximum-likelihood estimation*

Description

Runs `ssj_mle` over a grid of `maxsum` values and returns the solution with the lowest value of the selected information criterion. Unlike `auto_ssj_kfs` and `auto_ssj_atw`, `lambda` is not part of the grid: it is estimated by maximum likelihood inside every `ssj_mle` call, so that only `maxsum` has to be searched.

Usage

```
auto_ssj_mle(
  y,
  x,
  weights = NULL,
  ss_within = NULL,
  grid = seq(0, max(x) - min(x), length.out = 21L),
  ic = c("ebic", "bic", "hq", "aic", "aicc"),
  edf = TRUE,
  last_delta = 1,
  ebic_xi = 0.3
)
```

Arguments

`y`	numeric vector with the response (missing values allowed).
`x`	numeric vector with the covariate (no missing values allowed).
`weights`	optional numeric vector of positive observation weights, passed to the fitter; see its help page.
`ss_within`	optional pooled within-group sum of squares, passed to `ssj_mle`; see its help page.
`grid`	numeric vector of `maxsum` values to search; the default is 21 equally spaced values from 0 to the range of `x`.
`ic`	string with the information criterion used for the choice: the default is `"ebic"` (recommended when jumps are expected to be rare relative to the `n-1` candidate locations, see `ebic_xi`), but `"bic"`, `"hq"`, `"aic"` and `"aicc"` are also available.

edf	logical: passed to ssj_mle (default TRUE).
last_delta	numeric scalar: passed to ssj_mle (default 1).
ebic_xi	numeric scalar in [0,1] passed to ssj_mle: strength of the extended-BIC search penalty (default 0.3; only relevant when ic = "ebic"). See the "Choosing ebic_xi" section of ssj_mle for how it was calibrated.

Details

The pure smooth (maxsum = 0) is fitted first: its smoothed level disturbances are large exactly where the smooth model is "surprised" by a sudden level change, and are used to allocate the maxsum budget proportionally, giving the optimizer a warm start at the right locations. The search stops early when the criterion increases for three consecutive grid values.

Value

The output of the ssj_mle function corresponding to the best choice according to the selected information criterion.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

ssj_mle.

Examples

```
plot(faithful$eruptions, faithful$waiting)
of_mle <- auto_ssj_mle(y = faithful$waiting,
                       x = faithful$eruptions)
lines(x = of_mle$x, y = of_mle$smoothed_level, col = "darkgreen", lwd = 2)
```

BIC.hpj *BIC method for the class hpj*

Description

Extracts the Bayesian information criterion of one or more fitted HP filters with jumps.

Usage

```
## S3 method for class 'hpj'
BIC(object, ...)
```

Arguments

`object`	an object of class `hpj`.
`...`	optional additional objects of class `hpj`.

Value

If just one object is provided, a numeric value with the corresponding BIC. If multiple objects are provided, a `data.frame` with rows corresponding to the objects and columns representing the number of parameters in the model (`df`) and the BIC.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
BIC(hpj(y, 50), hpj(y, 100))
```

---

demand_temp *Electricity demand and temperature: the analysis sample*

Description

The homogeneous sub-sample of `load_temp` used in the electricity demand versus temperature application. It keeps only Tuesdays to Thursdays, the non-vacation months (February-March, May-July, September-November) and the central hours of the working day (10-15), and drops loads below 15000 MWh and temperatures of 35 degrees Celsius or above. These filters remove the weekday, holiday and time-of-day effects that would otherwise confound the relationship between demand and temperature.

Usage

```
data(demand_temp)
```

Format

A `data.frame` with 1691 rows and two columns:

Temperature The temperature of Milan, in degrees Celsius.

Load The electricity load of Northern Italy, in MWh.

Details

The script that builds this data set from `load_temp` is in `data-raw/demand_temp.R` of the package sources; `inst/scripts/load_temperature.R` shows how `load_temp` itself is assembled from the raw GME and Open-Meteo sources.

Source

GME, Open-Meteo.

See Also

`load_temp` for the full hourly series.

`dummyseas` *Seasonal dummy variables*

Description

Produces a matrix of seasonal dummies summing to zero, to be used as regressors. If s is the seasonal period, then the j-th dummy equals 1 in season j and $-1/(s-1)$ in the other seasons, so that the variable sums to 0 over one year.

Usage

```
dummyseas(n, s, remove = s)
```

Arguments

`n`	length of the time series.
`s`	seasonal period.
`remove`	index of the column to remove in order to avoid collinearity with the constant; by default it is column `s`. If `NULL`, no column is removed.

Value

A matrix with n rows and $s-1$ columns, unless `remove = NULL`, in which case it has s columns.

See Also

`trigseas` for the trigonometric alternative.

Examples

```
y <- log(AirPassengers)
X <- dummyseas(length(y), 12)
X <- cbind(X, t = seq_along(y))
reg <- lm(y ~ X)
```

employed_IT *Employed people in Italy by age class*

Description

Quarterly time series of the number of employed people by age class in Italy, over the period 1992Q4-2024Q1.

Usage

```
employed_IT
```

Format

A multivariate time series (class `ts`) with 126 rows and 10 columns:

Y15.24 Thousands of employed in the age class 15-24.

Y15.64 Thousands of employed in the age class 15-64.

Y20.64 Thousands of employed in the age class 20-64.

Y25.54 Thousands of employed in the age class 25-54.

Y55.64 Thousands of employed in the age class 55-64.

Y15.29 Thousands of employed in the age class 15-29.

Y15.74 Thousands of employed in the age class 15-74.

Y25.29 Thousands of employed in the age class 25-29.

Y29.54 Thousands of employed in the age class 29-54.

Y29.74 Thousands of employed in the age class 29-74.

Source

Eurostat.

hpfj *HP filter with automatic jump detection*

Description

Lower-level function for the Hodrick-Prescott filter with jumps. Most users should call `hpj` instead, unless they need more control and speed. Jumps happen contextually in the level and in the slope: the standard deviation of the slope disturbance is γ times the standard deviation of the level disturbance at time t. The HP smoothing parameter λ is estimated by maximum likelihood (assuming normally distributed disturbances) as in Wahba (1978): $\lambda = \sigma_\varepsilon^2/\sigma_\zeta^2$.

Usage

```
hpfj(y, maxsum = sd(y), edf = TRUE, parinit = NULL)
```

Arguments

y	numeric vector with the time series (missing values allowed).
maxsum	maximum sum of additional level standard deviations.
edf	logical: if TRUE (default) the effective degrees of freedom are computed, otherwise the number of degrees of freedom is computed in the LASSO-regression way.
parinit	either NULL or a numeric vector of 3+n starting values for the optimizer; the order of the parameters is sd(slope disturbance), sd(observation noise), square root of gamma, and n additional standard deviations for the level.

Value

A list with the following slots:

- opt: the output of the optimization function (nloptr).
- nobs: number of non-missing observations.
- df: model's degrees of freedom.
- maxsum: the maxsum argument.
- loglik: value of the log-likelihood at the maximum.
- pars: named vector (sigma_slope, sigma_noise, gamma).
- sigmas: additional standard deviations of the level disturbance.
- weights: effective-degrees-of-freedom weights, or NULL when edf = FALSE.
- ic: vector of information criteria (aic, aicc, bic, hq).
- smoothed_level: vector with the smoothed level with jumps (HP filter with jumps).
- var_smoothed_level: variance of the smoothed level.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Wahba, G. (1978). Improper priors, spline smoothing and the problem of guarding against model errors in regression. *Journal of the Royal Statistical Society. Series B*, **40**(3), 364–372. doi:10.1111/j.25176161.1978.tb01050.x

See Also

hpj, auto_hpfj, hpfj_fix.

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
plot(y, type = "l")
lines(mu, col = "blue")
hp <- hpfj(y, 60)
lines(hp$smoothed_level, col = "red")
```

---

hpfj_fix *HP filter with automatic jump detection and fixed smoothing constant*

Description

Lower-level function for the Hodrick-Prescott filter with jumps and a fixed smoothing parameter. Most users should call `hpj` instead, unless they need more control and speed. Jumps happen contextually in the level and in the slope: the standard deviation of the slope disturbance is γ times the standard deviation of the level disturbance at time t. Here the HP smoothing parameter λ is fixed by the user, so that $\sigma_\varepsilon^2 = \lambda\sigma_\zeta^2$.

Usage

```
hpfj_fix(y, lambda, maxsum = sd(y, na.rm = TRUE), edf = TRUE, parinit = NULL)
```

Arguments

`y`	numeric vector with the time series (missing values allowed).
`lambda`	either a numeric scalar with the smoothing constant, or a string with the frequency of the time series, one of `"daily"`, `"weekly"`, `"monthly"`, `"quarterly"`, `"annual"`; in the latter case the smoothing constant is computed according to Ravn and Uhlig (2002), that is $6.25s^4$, where s is the number of observations per year.
`maxsum`	maximum sum of additional level standard deviations.
`edf`	logical: if `TRUE` (default) the effective degrees of freedom are computed, otherwise the number of degrees of freedom is computed in the LASSO-regression way.
`parinit`	either `NULL` or a numeric vector of `2+n` starting values for the optimizer; the order of the parameters is sd(slope disturbance), square root of gamma, and `n` additional standard deviations for the level.

Value

A list with the following slots:

- `opt`: the output of the optimization function (`nloptr`).
- `nobs`: number of observations.
- `df`: model's degrees of freedom.
- `maxsum`: the `maxsum` argument.
- `loglik`: value of the log-likelihood at the maximum.
- `pars`: named vector (sigma_slope, sigma_noise, gamma).
- `sigmas`: additional standard deviations of the level disturbance.
- `weights`: effective-degrees-of-freedom weights, or `NULL` when `edf = FALSE`.
- `ic`: vector of information criteria (aic, aicc, bic, hq).
- `smoothed_level`: vector with the smoothed level with jumps (HP filter with jumps).
- `var_smoothed_level`: variance of the smoothed level.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Ravn, M. O. and Uhlig, H. (2002). On adjusting the Hodrick-Prescott filter for the frequency of observations. *The Review of Economics and Statistics*, **84**(2), 371–376. doi:10.1162/003465302317411604

See Also

`hpj`, `auto_hpfj_fix`, `hpfj`.

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
plot(y, type = "l")
lines(mu, col = "blue")
hp <- hpfj_fix(y, 60, 60)
lines(hp$smoothed_level, col = "red")
```

hpfjx *HP filter with jumps and regressors (still experimental)*

Description

Same as hpfj, but with the possibility of including regressors. The regressors should be zero-mean, so that the HP filter can still be interpreted as a mean value of the time series. Jumps happen contextually in the level and in the slope: the standard deviation of the slope disturbance is γ times the standard deviation of the level disturbance at time t. The HP smoothing parameter λ is estimated by maximum likelihood (assuming normally distributed disturbances) as in Wahba (1978): $\lambda = \sigma_\varepsilon^2/\sigma_\zeta^2$.

Usage

```
hpfjx(y, X, maxsum = sd(y), edf = TRUE, parinit = NULL)
```

Arguments

y	numeric vector with the time series (missing values allowed).
X	numeric matrix with the regressors in the columns.
maxsum	maximum sum of additional level standard deviations.
edf	logical: if TRUE (default) the effective degrees of freedom are computed, otherwise the number of degrees of freedom is computed in the LASSO-regression way.
parinit	either NULL or a numeric vector of 3+n+k starting values for the optimizer, where k is the number of regressors; the order of the parameters is sd(slope disturbance), sd(observation noise), square root of gamma, n additional standard deviations for the level, and k regression coefficients.

Details

This function is still experimental and needs more testing, since it does not always work as expected. For this reason the wrapper hpj does not allow regressors at the moment.

Value

A list with the following slots:

- opt: the output of the optimization function (nloptr).
- nobs: number of observations.
- df: model's degrees of freedom.
- maxsum: the maxsum argument.
- loglik: value of the log-likelihood at the maximum.
- coefficients: estimated regression coefficients.
- pars: named vector (sigma_slope, sigma_noise, gamma).

- `sigmas`: additional standard deviations of the level disturbance.
- `weights`: effective-degrees-of-freedom weights, or `NULL` when `edf = FALSE`.
- `ic`: vector of information criteria (aic, aicc, bic, hq).
- `smoothed_level`: vector with the smoothed level with jumps (HP filter with jumps).
- `var_smoothed_level`: variance of the smoothed level.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Wahba, G. (1978). Improper priors, spline smoothing and the problem of guarding against model errors in regression. *Journal of the Royal Statistical Society. Series B*, **40**(3), 364–372. doi:10.1111/j.25176161.1978.tb01050.x

See Also

`hpfj`, `auto_hpfjx`, `trigseas`, `dummyseas`.

Examples

```
y <- log(AirPassengers)
n <- length(y)
mod <- hpfjx(y, trigseas(n, 12))
trend <- ts(mod$smoothed_level, start(y), frequency = 12)
plot(y)
lines(trend, col = "red")
```

---

hpj *Hodrick-Prescott filter with jumps*

Description

Wrapper for the various Hodrick-Prescott filters with jumps. The end user should call this function: it is simpler and more flexible than the lower-level `hpfj*` functions.

Usage

```
hpj(
  y,
  maxsum = NULL,
  lambda = NULL,
  xreg = NULL,
  ic = c("bic", "hq", "aic", "aicc"),
  scl = 1000
)
```

Arguments

y	either a numeric vector or a time series object (ts, zoo, xts, timeSeries) containing the time series to filter.
maxsum	maximum sum of additional level standard deviations; if NULL (default), the value is selected automatically on a grid using the information criterion ic.
lambda	smoothing constant of the HP filter; if NULL (default), the value is estimated by maximum likelihood.
xreg	matrix of regressors. Not supported yet: passing a non-NULL value raises an error. Use hpfjx for the (still experimental) version with regressors.
ic	string with the information criterion used for the automatic choice of maxsum: the default is "bic" (simulations show this is the best choice), but "hq", "aic" and "aicc" are also available.
scl	scaling factor for the time series (default 1000): the series is rescaled as (y - min(y))/(max(y) - min(y))*scl, since the default starting values of the optimizer work well on this scale. If scl is the string "original", the series is not rescaled.

Value

An S3 object of class hpj with the following slots:

- y: the input time series.
- maxsum: the maximum sum of additional standard deviations.
- lambda: the smoothing constant of the HP filter.
- pars: vector of estimated parameters (sigma_slope, sigma_noise, gamma).
- hpj: the time series of the HP filter with jumps.
- hpj_std: the time series of the standard deviations of the HP filter with jumps.
- breaks: vector of indices of the detected breaks.
- sigmas: vector of additional standard deviations of the level disturbance.
- xreg: matrix of regressors (always NULL in this version).
- df: model's degrees of freedom.
- loglik: value of the log-likelihood at the maximum.
- ic: vector of information criteria (aic, aicc, bic, hq).
- opt: the output of the optimization function (nloptr).
- call: the call to the function.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

See Also

hpfj, hpfj_fix, plot.hpj, print.hpj.

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
hp <- hpj(y, 50)
print(hp)
plot(hp, use_ggplot = FALSE)
```

`llt` *Kalman filtering and smoothing for local linear trend plus noise*

Description

It uses the power of C++, scalar computation and pointers to run the Kalman filter, the smoother and compute the log-likelihood. The R user has to supply many vectors that in most cases will be overwritten by the llt() function since they are passed by reference. All passed parameters must be numerical (floating point) vectors: any other kind of variable may cause serious problems to the stability of your system. Passing vectors of integers will make the computations fail.

Usage

```
llt(
  y,
  var_eps,
  var_eta,
  var_zeta,
  cov_eta_zeta,
  a1,
  a2,
  p11,
  p12,
  p22,
  k1,
  k2,
  i,
  f,
  r1,
  r2,
  n11,
  n12,
  n22,
  e,
  d,
  w_ = NULL
)
```

Arguments

`y`	vector of n observations
`var_eps`	vector of n variances for the observation noises
`var_eta`	vector of n variances for the level disturbances
`var_zeta`	vector of n variances for the slope disturbances
`cov_eta_zeta`	vector of n covariances between level and slope disturbances
`a1`	vector of n+1 one-step-ahead prediction of the level; the first element is the initial condition for the level at time t=1, the other elements are arbitrary and will be overwritten
`a2`	vector of n+1 one-step-ahead prediction of the slope; the first element is the initial condition for the slope at time t=1, the other elements are arbitrary and will be overwritten
`p11`	vector of n+1 one-step-ahead prediction error variance of the level; the first element is the initial condition for the level at time t=1, the other elements are arbitrary and will be overwritten
`p12`	vector of n+1 one-step-ahead prediction covariances for level and slope; the first element is the initial condition for the slope at time t=1, the other elements are arbitrary and will be overwritten
`p22`	vector of n+1 one-step-ahead prediction error variance of the slope; the first element is the initial condition for the level at time t=1, the other elements are arbitrary and will be overwritten
`k1`	vector of the n Kalman gains for the level equation; values are arbitrary and will be overwritten;
`k2`	vector of the n Kalman gains for the slope equation; values are arbitrary and will be overwritten;
`i`	vector of the n innovations; values are arbitrary and will be overwritten;
`f`	vector of the n innovation variances; values are arbitrary and will be overwritten;
`r1`	vector of the n+1 smoothers (Th.5.4 in Pelagatti, 2015) for the level equation; values are arbitrary and will be overwritten;
`r2`	vector of the n+1 smoothers (Th.5.4 in Pelagatti, 2015) for the slope equation; values are arbitrary and will be overwritten;
`n11`	vector of the n+1 variance smoothers (Th.5.4 in Pelagatti, 2015) for the level equation; values are arbitrary and will be overwritten;
`n12`	vector of the n+1 covariance smoothers (Th.5.4 in Pelagatti, 2015) for the level and slope; values are arbitrary and will be overwritten;
`n22`	vector of the n+1 variance smoothers (Th.5.4 in Pelagatti, 2015) for the slope equation; values are arbitrary and will be overwritten;
`e`	vector of the n observation error smoothers (Th.5.4 in Pelagatti, 2015); values are arbitrary and will be overwritten;
`d`	vector of the n observation error variance smoothers (Th.5.4 in Pelagatti, 2015); values are arbitrary and will be overwritten;

w_ NULL (default) or vector of n weights, where element t is the weight of observation y_t on the smoothed signal at time t, that is the t-th diagonal element of the smoothing (hat) matrix, computed with the recursion of Koopman and Harvey (2003); their sum is the effective degrees of freedom. A missing y_t receives weight zero. Values are arbitrary on entry and will be overwritten;

Value

The value of the Gaussian log-likelihood net of the -log(2*pi)*n/2 part that can be added if needed.

---

load_temp *Hourly electricity load of Northern Italy and temperature of Milan*

Description

Hourly observations from 2023-01-01 01:00:00 to 2025-09-30 23:00:00. The electricity load for Northern Italy was downloaded from the FTP site of the GME (mercatoelettrico.org). The temperature was downloaded from the Open-Meteo archive (latitude 45.4642, longitude 9.1900).

Usage

```
data(load_temp)
```

Format

A `data.frame` with 24092 rows and three columns:

Datetime A `POSIXct` object with day and hour.

Load The electricity load of Northern Italy, in MWh.

Temperature The temperature of Milan, in degrees Celsius.

Source

GME, Open-Meteo.

logLik.hpj *logLik method for the class hpj*

Description

Extracts the Gaussian log-likelihood of a fitted HP filter with jumps.

Usage

```
## S3 method for class 'hpj'
logLik(object, ...)
```

Arguments

object	an object of class hpj.
...	not used: for consistency with the generic function.

Value

An object of class logLik with the log-likelihood value and two attributes: df, the number of degrees of freedom, and nobs, the number of observations.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
logLik(hpj(y, 50))
```

mse *Mean squared error*

Description

Computes the mean squared difference between the elements of the vectors y and x.

Usage

```
mse(y, x)
```

Arguments

y	numeric vector.
x	numeric vector of the same length as y.

Value

The scalar mean squared difference (error).

Examples

```
mse(1:3, 2:4)
```

---

nobs.hpj *nobs method for the class hpj*

Description

Extracts the number of non-missing observations used to fit an HP filter with jumps.

Usage

```
## S3 method for class 'hpj'
nobs(object, ...)
```

Arguments

object	an object of class hpj.
...	not used: for consistency with the generic function.

Value

The number of non-missing observations of the time series to which the HP filter with jumps has been applied.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
nobs(hpj(y, 50))
```

`plot.hpj` *Plot method for the class hpj*

Description

Plots the original time series together with the estimated Hodrick-Prescott filter with jumps, optionally with a confidence band and vertical lines at the detected breaks.

Usage

```
## S3 method for class 'hpj'
plot(
  x,
  prob = NULL,
  show_breaks = TRUE,
  main = "original + filter",
  use_ggplot = TRUE,
  ...
)
```

Arguments

`x`	an object of class `hpj`.
`prob`	coverage probability of the confidence interval for the filter; if `NULL` (default) no interval is drawn.
`show_breaks`	logical: if `TRUE` (default), vertical lines are drawn at the detected breaks.
`main`	title of the plot.
`use_ggplot`	logical: if `TRUE` (default), the plot is produced with **ggplot2**, which must be installed; otherwise base graphics are used.
`...`	additional arguments passed to the base `plot` function (only used when `use_ggplot = FALSE`).

Value

If `use_ggplot` is `TRUE`, a **ggplot2** object; otherwise a base plot is drawn and `NULL` is returned invisibly.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
hp <- hpj(y, 50)
plot(hp, prob = 0.95, use_ggplot = FALSE)
```

`plot.ssj` *Plot method for the class ssj*

Description

Draws the observed data with the fitted smoothing spline with jumps and, optionally, a confidence band and vertical lines at the declared discontinuities. For the domain-warping estimators a second panel below shows the estimated warps $\hat{\gamma}_t$, which measure the strength of each jump, together with the detection threshold. This reproduces Figure 4 of Pelagatti, Maranzano and Gatto Ronchieri (2026).

Usage

```
## S3 method for class 'ssj'
plot(
  x,
  y = NULL,
  prob = NULL,
  show_breaks = TRUE,
  show_warp = TRUE,
  warp_frac = 0.3,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  col_fit = "firebrick",
  col_data = "steelblue",
  use_ggplot = TRUE,
  ...
)
```

Arguments

`x`	an object of class `ssj`.
`y`	not used: for consistency with the generic function.
`prob`	coverage probability of the confidence band around the fitted curve; if `NULL` (default), or if the estimator does not return pointwise variances, no band is drawn.

show_breaks	logical: if TRUE (default), vertical dashed lines are drawn at the declared discontinuities.
show_warp	logical: if TRUE (default), a lower panel with the estimated domain warps is added, when the estimator provides them.
warp_frac	share of the plotting height given to the lower panel, a number strictly between 0 and 1 (default 0.3). The warp panel carries one value per interval and most of them are zero, so it needs much less room than the fitted curve; 0.5, which is what equal panels amount to, wastes half the figure. It has no effect when there is no lower panel. In the **ggplot2** figure it requires **ggplot2** 4.0.0 or later, the version that introduced the panel.heights theme element; earlier versions draw panels of equal height and a non-default warp_frac then warns. Base graphics honour it at any version.
main	title of the plot; the default, NULL, draws no title.
xlab, ylab	axis labels; if NULL defaults are used.
col_fit, col_data	colours of the fitted curve and of the observations.
use_ggplot	logical: if TRUE (the default, as in plot.hpj) the figure is built with **ggplot2** and *returned* rather than drawn, so it can be further modified, themed, saved with ggplot2::ggsave or combined with other plots. Set it to FALSE for base graphics, which draws directly and returns NULL. If **ggplot2** is not installed the argument is ignored with a warning and base graphics are used. Note the consequence for scripts: with the default the plot is a value, so at the top level of a script or inside a loop it must be printed to appear, exactly as any **ggplot2** figure must.
...	further arguments passed to ssj_breaks, such as thresh, frac_total or frac_delta.

Value

With the default use_ggplot = TRUE, a ggplot object; the fitted curve and the warp profile are facets of a single plot rather than two separate figures, so that one value is returned and the two x axes stay aligned. With use_ggplot = FALSE, NULL invisibly, the function being called for its side effect of drawing.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

print.ssj, ssj_breaks.

Examples

```
# 'maxsum' is a warping budget in the units of x; the range of eruptions
# is about 3.5, so a budget of 1 is already generous.
fit <- ssj_atw(y = faithful$waiting, x = faithful$eruptions,
```

```
               lambda = 1, maxsum = 1)

# the default figure; no title is drawn unless one is asked for
plot(fit)

# base graphics instead.  They honour 'warp_frac' at any ggplot2 version,
# so give the warp panel a fifth of the height rather than the default 30%
plot(fit, use_ggplot = FALSE, warp_frac = 0.2)

# the ggplot2 figure is returned rather than drawn, so a single call can
# carry the title, the axis labels and an absolute threshold on the warps
# (as in Section 4.2 of the paper), and the result still takes any further
# ggplot2 layer before it is printed
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(fit, thresh = 1, main = "Old Faithful",
       xlab = "eruption length", ylab = "waiting time") +
    ggplot2::labs(caption = "data: faithful")
}
```

`predict.ssj` *Predict from a smoothing spline with jumps*

Description

Evaluates a fitted `ssj` object at arbitrary locations. For the domain-warping estimators this is the cubic the model actually implies, not an interpolation of the fitted values.

Usage

```
## S3 method for class 'ssj'
predict(object, xnew = NULL, ...)
```

Arguments

`object`	an object of class `ssj`.
`xnew`	numeric vector of locations at which to evaluate the fit; if `NULL` (default), the fitted values at the sampling sites are returned.
`...`	passed to `predict_cssd_dp` for CSSD objects, otherwise ignored.

Details

The state of the warping model is the pair $(f, df/ds)$ at every sampling site, where s is the latent (warped) coordinate. Between two sites the fitted function is therefore a cubic in s, and since s is a linear dilation of x on each interval, it is a cubic in x as well. Cubic Hermite interpolation with the stored levels and slopes reproduces it exactly. On an interval carrying a large warp the same cubic is traversed over a short physical distance, which is what makes the transition look like a jump; a linear interpolation of the fitted values would cut that corner, and a natural cubic spline through the fitted values alone would smooth it away.

Outside the range of the data the fit is extended linearly from the end knot, rather than by extrapolating the end cubic.

For an object produced by ssj_cssd_dp the call is passed to predict_cssd_dp; for one produced by ssj_cssd, whose fit is piecewise constant by construction, a step interpolation is used.

Value

Numeric vector of fitted values, in the same order as xnew.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

plot.ssj, ssj_breaks.

Examples

```
set.seed(2026)
x <- sort(runif(80))
y <- sin(4*pi*x) + (x > 0.5) + rnorm(80, sd = 0.1)
# a short grid keeps the example quick; the default 21-point one is finer
fit <- auto_ssj_mle(y, x, ic = "ebic", ebic_xi = 0.1,
                    grid = seq(0, 1, length.out = 9))

xg <- seq(0, 1, length.out = 400)
plot(x, y, pch = 16, cex = 0.6, col = "grey60")
lines(xg, predict(fit, xg), col = "firebrick", lwd = 2)

# the cubic keeps the transition sharp where a linear interpolation of the
# fitted values would cut the corner
lines(fit$x, fit$smoothed_level, col = "steelblue", lty = 2)
```

---

predict_cssd_dp *Predict from a fitted CSSD at new locations*

Description

Evaluates the piecewise cubic spline returned by ssj_cssd_dp at arbitrary query points. Query points outside the range of the data are extrapolated by the cubic polynomial of the nearest segment.

Usage

```
predict_cssd_dp(fit, xnew, y_orig = NULL, x_orig = NULL, delta = NULL)
```

Arguments

`fit`	object returned by `ssj_cssd_dp`.
`xnew`	numeric vector of query locations (it need not be sorted).
`y_orig, x_orig`	original data vectors, needed to re-fit the Hermite control points on each segment. If `NULL` (default), the sorted data stored in `fit` by `ssj_cssd_dp` are used.
`delta`	positive scalar or vector with the noise standard deviations (the same as those used to obtain `fit`). If `NULL` (default), the value stored in `fit` is used.

Value

Numeric vector of fitted values, in the same order as `xnew`.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

`ssj_cssd_dp`.

Examples

```
set.seed(1)
x    <- sort(runif(100))
y    <- sin(4 * pi * x) + c(rep(0, 50), rep(1, 50)) + rnorm(100, sd = 0.1)
fit  <- ssj_cssd_dp(y, x, p = 0.999, gamma = 4)
xnew <- seq(0, 1, length.out = 200)
plot(x, y)
lines(xnew, predict_cssd_dp(fit, xnew), col = "darkgreen", lwd = 2)
```

`print.hpj` *Print method for the class hpj*

Description

Prints the call, the estimated parameters, the information criteria and the dates of the detected breaks.

Usage

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

Arguments

x	an object of class hpj.
...	not used: for consistency with the generic function.

Value

x, invisibly. Called for its side effect of printing.

References

Maranzano, P. and Pelagatti, M. (2025). A Hodrick-Prescott filter with automatically selected breaks. *Economic Modelling*, **150**, 107132. doi:10.1016/j.econmod.2025.107132

Examples

```
set.seed(202311)
n <- 100
mu <- 100*cos(3*pi/n*(1:n)) - ((1:n) > 50)*n - c(rep(0, 50), 1:50)*10
y <- mu + rnorm(n, sd = 20)
print(hpj(y, 50))
```

---

print.ssj *Print method for the class ssj*

Description

Prints the estimation method, the call, the estimated parameters, the information criteria and a table of the declared discontinuities.

Usage

```
## S3 method for class 'ssj'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	an object of class ssj.
digits	number of significant digits used for the numbers printed.
...	further arguments passed to ssj_breaks, such as thresh, frac_total or frac_delta.

Value

x, invisibly. Called for its side effect of printing.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

plot.ssj, ssj_breaks.

Examples

```
# 'maxsum' is a warping budget in the units of x; the range of eruptions
# is about 3.5, so a budget of 1 is already generous.
fit <- ssj_atw(y = faithful$waiting, x = faithful$eruptions,
               lambda = 1, maxsum = 1)
print(fit)
```

`ssj_atw` *Smoothing splines with automatic jump detection using the alternating time-warping algorithm*

Description

Same model as ssj_fix, but fitted with the alternating time-warping (ATW) algorithm: a two-step block coordinate descent in which the f-step solves a banded linear system in $O(n)$ operations and the γ-step performs a gradient descent on the roughness energy followed by a projection onto the simplex $\{\gamma \geq 0, \sum_i \gamma_i \leq M\}$.

Usage

```
ssj_atw(
  y,
  x,
  lambda,
  maxsum = max(x) - min(x),
  weights = NULL,
  max_iter = 100,
  tol = 1e-04,
  learning_rate = 0.1,
  gamma_init = numeric(0),
  ebic_xi = 0.25
)
```

Arguments

y	numeric vector with the response (no missing values allowed).
x	numeric vector with the covariate (no missing values allowed).
lambda	numeric scalar with the smoothing parameter.
maxsum	maximum sum of the γ_t, that is the warping budget M of the constraint $\sum_t \gamma_t \leq M$. It is expressed in the units of x and defaults to the range of x, the same upper end as the grid searched by auto_ssj_atw.
weights	optional numeric vector of positive observation weights, one per element of y. The data term becomes $\sum_i w_i(y_i - f_i)^2$, so a point with weight w_i counts as the mean of w_i independent observations, and the information criteria use $\sum_i w_i$ in place of the number of design points. Weights are averaged along with y when tied values of x are collapsed. The default, NULL, means unit weights.
max_iter	maximum number of iterations (default 100).
tol	tolerance for numerical convergence (default 1e-4).
learning_rate	initial and fallback step size for the Barzilai-Borwein adaptive scheme (default 0.1).
gamma_init	optional numeric vector of length n-1 with a warm start for γ; ignored when its length does not match n-1.
ebic_xi	numeric scalar in [0,1]: strength of the extended-BIC penalty for searching over the n-1 candidate jump locations (default 0.25; 0 reduces ebic to plain bic). The default is calibrated rather than taken from theory; see the note below.

Details

Internally x is mapped to $[0, 1]$ and y is scaled to unit variance, so that lambda and maxsum are interpreted in the original units of the data. Tied values of x are collapsed by averaging the corresponding y; the fitted values are therefore returned at the (possibly shorter) vector of unique, sorted x values available in the x slot of the output.

Value

An object of class ssj (a list, so that all the slots below remain directly accessible) with a print and a plot method. Its slots are:

- nobs: number of distinct sampling sites used.
- df: model's effective degrees of freedom.
- maxsum: the maxsum argument.
- pars: named vector (sigma, lambda), where sigma is the unbiased residual standard deviation $\sqrt{rss/(n - edf)}$ that the information criteria are evaluated at, and lambda is the argument, recorded so that it can be recovered from a fit. The residual degrees of freedom are one per fitted point, n, and not one per underlying observation: the weighted rss holds only between-point variation, the within-group scatter of tied x having been removed by the collapse.
- loglik: Gaussian log-likelihood of the n fitted points, evaluated at the unbiased residual variance $\sigma^2 = rss/(n - edf)$, on the scale of y as supplied.
- gcv: generalized cross-validation criterion.

- gamma: vector of additional times (time warps) of length nobs - 1.
- ic: vector of information criteria (aic, aicc, bic, hq, ebic).
- smoothed_level: vector with the smoothing spline with jumps.
- x: the ordered vector of distinct x values.

Choosing ebic_xi

The default is the value that minimises worst-case regret in root mean squared error over the 48-cell design of inst/scripts/xi_calibration.R. Unlike ssj_mle and ssj_kfs, whose optima are flat in n, the best value for this estimator falls steadily with the sample size – about 0.32, 0.22, 0.15 and 0.12 at n = 50, 100, 200 and 400. What is constant instead is the total charge per declared discontinuity, $s \log n + 2\xi \log(n-1)$ with $s \approx 0.42$ the degrees of freedom each activated warp adds, which held at 3.9 across the design (trend -0.10 per $\log n$, 95% interval $[-0.23, 0.02]$). That gives the rule

$$\xi(n) = \frac{3.93 - 0.419 \log n}{2 \log(n-1)},$$

which is worth using in place of the constant when n is far from a few hundred: at $n = 400$ it improved root mean squared error by 4.4% over the fixed 0.25. The constant is the default because it is the value that behaves acceptably everywhere, and because a default that changes with the data is easy to overlook.

Note that the charge of about 3.9 on the $-2 \log L$ scale is close to 4, that is roughly two log-likelihood units per discontinuity – what AIC charges for one free parameter.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

Chen, J. and Chen, Z. (2008). Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, **95**(3), 759–771. doi:10.1093/biomet/asn034

See Also

auto_ssj_atw, ssj_fix, ssj_kfs, ssj_mle.

Examples

```
# 'maxsum' is a warping budget in the units of x; the range of eruptions
# is about 3.5, so a budget of 1 is already generous.
plot(faithful$eruptions, faithful$waiting)
of_jum <- ssj_atw(y = faithful$waiting,
                  x = faithful$eruptions,
                  lambda = 1,
                  maxsum = 1)
lines(x = of_jum$x, y = of_jum$smoothed_level, col = "red", lwd = 2)
```

`ssj_breaks` *Discontinuities declared by a smoothing spline with jumps*

Description

Applies the detection rule of Pelagatti, Maranzano and Gatto Ronchieri (2026) to a fitted object of class `ssj` and returns the declared discontinuities, their locations, the corresponding domain warps and the size of the jump in the fitted curve.

Usage

```
ssj_breaks(object, frac_total = 0.05, frac_delta = 0.3, thresh = NULL)
```

Arguments

`object`	an object of class `ssj`.
`frac_total`	minimum share of the total warping that an interval must carry to be declared a discontinuity (default 0.05).
`frac_delta`	minimum multiple of the mean sampling interval that the warp must reach (default 0.30).
`thresh`	optional absolute threshold on the warp; when supplied it overrides `frac_total` and `frac_delta`.

Details

For the domain-warping estimators (`ssj_mle`, `ssj_kfs`, `ssj_atw`) an interval is declared a discontinuity when the estimated warp $\hat{\gamma}_t$ clears *both* `frac_total` times the total warping in the sample and `frac_delta` times the mean sampling interval. The first condition filters out numerically small, spurious warps, the second keeps the rule meaningful in the units of `x`; together they retain genuine jumps of very different magnitudes. Supplying `thresh` replaces both conditions by the single rule $\hat{\gamma}_t >$ `thresh`, which is convenient with equally spaced data, where one unit of the domain has a direct interpretation (a week, say) and `thresh = 1` means that the algorithm has inserted at least one extra unit of fictitious domain length.

Since γ_t lives on the interval between x_t and x_{t+1}, a break is dated at x_{t+1}, the first observation belonging to the new regime.

For `ssj_fix`, which parameterises jumps through time-varying disturbance standard deviations rather than domain warps, only the `frac_total` rule is applied, because the `frac_delta` condition has no meaning in those units. For the CSSD estimators the discontinuity set is part of the solution and is returned as it is, ignoring the thresholds.

Value

A `data.frame`, with one row per declared discontinuity, and columns:

- `index`: index of the first observation of the new regime.
- `x`: its location, that is the date of the break.

- `warp`: the estimated domain warp $\hat{\gamma}_t$ (or the additional standard deviation for `ssj_fix`, `NA` for the CSSD estimators).
- `jump`: the size of the jump, that is the change of the fitted curve across the break.

It has zero rows when no discontinuity is declared.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

`plot.ssj`, `print.ssj`.

Examples

```
# 'maxsum' is a warping budget in the units of x; the range of eruptions
# is about 3.5, so a budget of 1 is already generous.
fit <- ssj_atw(y = faithful$waiting, x = faithful$eruptions,
               lambda = 1, maxsum = 1)
ssj_breaks(fit)
```

`ssj_cssd` *Total-variation denoising for irregularly sampled signals*

Description

Solves the total-variation denoising problem

$$\min_u \frac{1}{2}\|y-u\|_2^2 + \lambda\|Du\|_1,$$

where D is the unweighted first-difference operator $(Du)_i = u_{i+1} - u_i$, using ADMM with a pre-factorized $I + \rho D^\top D$ linear system. The minimizer u is piecewise constant, with non-zero first differences at the detected jump locations.

Usage

```
ssj_cssd(y, x = NULL, lambda, max_iter = 100L, rho = 1)
```

Arguments

`y`	numeric vector of observations (no missing values).
`x`	numeric vector of sampling locations (no missing values; it need not be sorted). If `NULL` (default), regular spacing `1:n` is assumed.

`lambda`	positive regularization parameter controlling the soft-threshold λ/ρ applied to the first differences of u. Set it strictly above the noise floor $\sigma\sqrt{2}$ and below the smallest expected jump amplitude. When the variation of the smooth signal overlaps the jumps, select it by cross-validation.
`max_iter`	maximum number of ADMM iterations (default 100).
`rho`	positive augmented-Lagrangian step size; it controls the convergence speed but not the solution (default 1.0).

Details

This implements the simplified total-variation (first-order) variant related to the CSSD framework of Storath and Weinmann (2024). The full CSSD (piecewise cubic smoothing spline with discontinuities) is implemented in `ssj_cssd_dp`; the total-variation formulation used here replaces the C^2 roughness penalty with an ℓ_1 penalty on first differences and is therefore much faster, at the cost of a piecewise constant (rather than piecewise cubic) fit.

The unweighted D (entries ± 1) is used rather than the slope-based $(Du)_i = (u_{i+1} - u_i)/\delta_i$: the $1/\delta$ weighting inflates the eigenvalues of $D^\top D$ by $1/h^2 \sim 10^4$ for $h \sim 0.01$, which drives the ADMM u-update towards zero in the first iteration and stalls it permanently.

Value

An object of class `ssj` (a list, so that all the slots below remain directly accessible) with a `print` and a `plot` method. Its slots are:

- `smoothed_level`: fitted piecewise-constant signal at each (sorted) sampling site.
- `x`: sorted sampling sites.
- `nobs`: number of distinct sampling sites used.
- `lambda`: the `lambda` value used.
- `gamma`: absolute first differences $|u_{i+1} - u_i|$ of the fit; entries greater than zero mark the detected jump locations. It has length `nobs - 1` and is named by the midpoints $(x_i + x_{i+1})/2$.

References

Storath, M. and Weinmann, A. (2024). Smoothing splines for discontinuous signals. *Journal of Computational and Graphical Statistics*, **33**(2), 651–664. doi:10.1080/10618600.2023.2262000

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

`ssj_cssd_dp` for the exact cubic CSSD.

Examples

```
set.seed(1)
x   <- sort(runif(100))
y   <- sin(4 * pi * x) + c(rep(0, 50), rep(1, 50)) + rnorm(100, sd = 0.1)
fit <- ssj_cssd(y, x, lambda = 5)
```

```
plot(x, y)
lines(fit$x, fit$smoothed_level, col = "steelblue", lwd = 2)
```

`ssj_cssd_dp` *Cubic smoothing spline with discontinuities (CSSD)*

Description

Computes the global minimiser of the weak-rod model

$$\min_{f,J} p \sum_i \left(\frac{y_i - f(x_i)}{\delta_i} \right)^2 + (1-p) \int (f''(t))^2 \, dt + \gamma |J|$$

using dynamic programming with incremental QR updates (Givens rotations), exactly as described in Storath and Weinmann (2024).

Usage

```
ssj_cssd_dp(y, x = NULL, p = 0.999, gamma = 1, delta = 1)
```

Arguments

`y`	numeric vector of observations (no missing values).
`x`	numeric vector of sampling locations (no missing values; it is sorted internally). If `NULL` (default), regular spacing `1:n` is assumed.
`p`	stiffness parameter in $(0, 1)$: weight of the data-fit term relative to the smoothness penalty. Values close to 1 favour the data fit; values close to 0 favour smoothness (approaching piecewise-linear regression as $p \to 0$). Use `auto_ssj_cssd` for automatic selection by K-fold cross-validation.
`gamma`	jump-count penalty $\gamma > 0$. Larger values discourage discontinuities; $\gamma \to \infty$ recovers the classical cubic smoothing spline. Use `auto_ssj_cssd` for automatic selection.
`delta`	positive scalar or vector of length `n` with the pointwise noise standard deviations δ_i. Default 1 (all equal).

Details

The algorithm computes the energy of every candidate interval, so both its running time and its memory footprint grow as $O(n^2)$. Sample sizes of a few thousand observations are the practical limit.

Value

An object of class `ssj` (a list, so that all the slots below remain directly accessible) with a `print` and a `plot` method. Its slots are:

- `smoothed_level`: fitted cubic spline values at each (sorted) data site.
- `x`: sorted data sites.
- `y`: the response, sorted (and averaged over ties) to match `x`.
- `disc_locs`: x-coordinates of the detected discontinuities (midpoints between adjacent data sites).
- `n_disc`: number of detected discontinuities.
- `p`, `gamma`: the parameters used.
- `delta`: the vector of noise standard deviations used.
- `nobs`: number of distinct data sites used.

References

Storath, M. and Weinmann, A. (2024). Smoothing splines for discontinuous signals. *Journal of Computational and Graphical Statistics*, **33**(2), 651–664. doi:10.1080/10618600.2023.2262000

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

`auto_ssj_cssd`, `predict_cssd_dp`, `ssj_cssd`.

Examples

```
set.seed(1)
x   <- sort(runif(100))
y   <- sin(4 * pi * x) + c(rep(0, 50), rep(1, 50)) + rnorm(100, sd = 0.1)
fit <- ssj_cssd_dp(y, x, p = 0.999, gamma = 4)
plot(x, y)
lines(fit$x, fit$smoothed_level, col = "darkgreen", lwd = 2)
```

---

ssj_fix *Smoothing splines with automatic jump detection and fixed smoothing parameter*

Description

Lower-level function for smoothing splines with jumps and a fixed smoothing parameter. The additional variance of the level disturbance at each sampling site is estimated by constrained maximum likelihood, using the Kalman filter and smoother on the state-space form of the cubic smoothing spline.

Usage

```
ssj_fix(
  y,
  x,
  lambda,
  maxsum = sd(y, na.rm = TRUE)/mean(diff(x)),
  weights = NULL,
  edf = TRUE,
  parinit = NULL,
  last_delta = 1
)
```

Arguments

y	numeric vector with the response (missing values allowed).
x	numeric vector with the covariate (no missing values allowed).
lambda	numeric scalar with the smoothing parameter.
maxsum	maximum sum of the additional standard deviations. Note that this parameterises the jumps through the disturbance variances, so, unlike the budget of ssj_mle, ssj_kfs and ssj_atw, it is *not* expressed in the units of x.
weights	optional numeric vector of positive observation weights, one per element of y. The observation variance becomes $H_t = \lambda\sigma^2/w_t$, so a point carrying weight w_t is treated as the mean of w_t independent observations. This is what makes it correct to collapse tied values of x to their mean and pass the number of replicates as the weight: the Gaussian likelihood factorises, so the weighted fit on the collapsed data is identical to the fit on the raw replicates. The default, NULL, means unit weights.
edf	logical: if TRUE (default) the effective degrees of freedom are computed, otherwise the number of degrees of freedom is computed in the LASSO-regression way.
parinit	either NULL or a numeric vector of 1+n starting values for the optimizer; the order of the parameters is the constant standard deviation followed by n additional standard deviations.
last_delta	numeric scalar with the difference $x_{n+1} - x_n$; it is not relevant for most applications and defaults to 1.

Details

Tied values of x are collapsed to their weighted mean before fitting, and the fitted values are returned at the vector of unique, sorted x available in the x slot of the output. This keeps the problem well posed: with ties the interval between two replicates has zero length, and a warp placed there would be a discontinuity between two observations recorded at the same x, which is not identified. The replicate counts become observation weights automatically, so no information is lost.

Value

An object of class `ssj` (a list, so that all the slots below remain directly accessible) with a `print` and a `plot` method. Its slots are:

- `opt`: the output of the optimization function (`nloptr`).
- `nobs`: number of observations.
- `df`: model's degrees of freedom.
- `maxsum`: the `maxsum` argument.
- `loglik`: value of the log-likelihood at the maximum.
- `pars`: named vector (sigma, lambda).
- `sigmas`: additional standard deviations of the level disturbance.
- `weights`: effective-degrees-of-freedom weights, or `NULL` when `edf = FALSE`.
- `ic`: vector of information criteria (aic, aicc, bic, hq).
- `smoothed_level`: vector with the smoothing spline with jumps.
- `var_smoothed_level`: vector with the variances of the smoothing spline.
- `x`: the ordered `x` variable.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

See Also

`auto_ssj_fix`, `ssj_kfs`, `ssj_mle`, `ssj_atw`.

Examples

```
plot(faithful$eruptions, faithful$waiting)
of_jum <- ssj_fix(y = faithful$waiting,
                  x = faithful$eruptions,
                  lambda = 1,
                  maxsum = 150)
lines(x = of_jum$x, y = of_jum$smoothed_level, col = "red", lwd = 2)
```

`ssj_kfs` *Smoothing splines with automatic jump detection via time warping and Kalman filter-smoother estimation*

Description

Implements the state-space time-warping model described in the companion paper. The effective inter-observation interval $\tau_t = \delta_t + \gamma_t$ replaces the fixed spacing in both the transition matrix and the process-noise covariance, so that jump detection reduces to finding the optimal γ by maximum likelihood.

Usage

```
ssj_kfs(
  y,
  x,
  lambda,
  maxsum = max(x) - min(x),
  weights = NULL,
  edf = TRUE,
  parinit = NULL,
  last_delta = 1,
  ebic_xi = 0.15
)
```

Arguments

`y`	numeric vector with the response (missing values allowed).
`x`	numeric vector with the covariate (no missing values allowed).
`lambda`	positive scalar smoothing parameter (ratio of observation to process noise variance).
`maxsum`	maximum sum of the γ_t, that is the warping budget M of the constraint $\sum_t \gamma_t \leq M$. It is expressed in the units of `x` and defaults to the range of `x`, the same upper end as the grid searched by `auto_ssj_kfs`.
`weights`	optional numeric vector of positive observation weights, one per element of `y`. The observation variance becomes $H_t = \lambda\sigma^2/w_t$, so a point carrying weight w_t is treated as the mean of w_t independent observations. This is what makes it correct to collapse tied values of `x` to their mean and pass the number of replicates as the weight: the Gaussian likelihood factorises, so the weighted fit on the collapsed data is identical to the fit on the raw replicates. The default, `NULL`, means unit weights.
`edf`	logical: if `TRUE` (default) the effective degrees of freedom are computed as the trace of the hat matrix; if `FALSE` the non-zero γ_t are counted.
`parinit`	`NULL` or a numeric vector of length `n` with starting values for the optimizer, namely $(\sigma, \gamma_1, \ldots, \gamma_{n-1})$; pass the previous solution for warm starting.

`last_delta` positive scalar: dummy spacing appended after the last observation (it does not affect the fit; default 1).

`ebic_xi` numeric scalar in `[0,1]`: strength of the extended-BIC penalty for searching over the `n-1` candidate jump locations (default 0.15; 0 reduces `ebic` to plain `bic`). The default is calibrated rather than taken from theory; see the note below.

Details

The analytical gradient of the log-likelihood with respect to γ_t is obtained through the Koopman-Shephard (2003) score formula applied to the Kalman smoother output, plus an extra term accounting for γ_t entering the transition matrix. The optimisation is carried out with the gradient-based AUGLAG/LBFGS algorithms of the NLopt library, with a fallback to CCSAQ whenever the KKT conditions are not satisfied at the reported optimum.

Tied values of `x` are collapsed to their weighted mean before fitting, and the fitted values are returned at the vector of unique, sorted `x` available in the `x` slot of the output. This keeps the problem well posed: with ties the interval between two replicates has zero length, and a warp placed there would be a discontinuity between two observations recorded at the same `x`, which is not identified. The replicate counts become observation weights automatically, so no information is lost.

Value

An object of class `ssj` (a list, so that all the slots below remain directly accessible) with a `print` and a `plot` method. Its slots are:

- `opt`: the output of the `nloptr` optimizer.
- `nobs`: number of observations.
- `df`: model's degrees of freedom.
- `maxsum`: the `maxsum` argument.
- `loglik`: Gaussian log-likelihood at the optimum, on the scale of `y` as supplied. Note that the diffuse initial conditions are approximated by a large proper prior, so the absolute value carries an additive constant set by that convention; differences across `maxsum` for one data set, which is what the information criteria compare, are unaffected.
- `pars`: named vector (sigma, lambda).
- `gamma`: estimated time-warp vector of length `n-1`.
- `ic`: vector of information criteria (aic, aicc, bic, hq, ebic).
- `smoothed_level`: smoothed estimate of f at each `x`.
- `var_smoothed_level`: pointwise variance of the smoothed level.
- `x`: the ordered `x` variable.

Choosing `ebic_xi`

The default is the value that minimises worst-case regret in root mean squared error over the 48-cell design of `inst/scripts/xi_calibration.R`. It is lower than the 0.3 used by `ssj_mle`, and the reason is not that the models differ – at a given `lambda` and budget they are the same fit – but that the procedures do. `auto_ssj_kfs` holds `lambda` fixed across the budget grid, whereas `auto_ssj_mle`

re-estimates it at every point; being able to re-tune the smoothness as the budget grows gives more scope to overfit, and so calls for a heavier penalty.

The value sits below the Chen and Chen (2008) consistency floor of about 0.5 because the plain BIC term already charges for each activated warp through `df`, at about $0.34 \log n$ per declared jump for this estimator – an implicit charge of roughly 0.17 on the ξ scale.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

Chen, J. and Chen, Z. (2008). Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, **95**(3), 759–771. doi:10.1093/biomet/asn034

Koopman, S. J. and Shephard, N. (1992). Exact score for time series models in state space form. *Biometrika*, **79**(4), 823–826. doi:10.1093/biomet/79.4.823

See Also

`auto_ssj_kfs`, `ssj_mle`, `ssj_atw`, `ssj_fix`.

Examples

```
# 'maxsum' is a warping budget in the units of x; the range of eruptions
# is about 3.5, so a budget of 1 is already generous.
plot(faithful$eruptions, faithful$waiting)
of_kfs <- ssj_kfs(y = faithful$waiting,
                  x = faithful$eruptions,
                  lambda = 1,
                  maxsum = 1)
lines(x = of_kfs$x, y = of_kfs$smoothed_level, col = "blue", lwd = 2)
```

---

ssj_mle *Smoothing splines with automatic jump detection via time warping and maximum-likelihood estimation of the smoothing parameter*

Description

Identical to `ssj_kfs`, except that `lambda` is no longer a fixed input: it is a free parameter optimized jointly with σ and γ. This avoids having to grid-search `lambda` (which would suffer from the same degrees-of-freedom problem as `maxsum`) on top of the `maxsum` grid search.

Usage

```
ssj_mle(
  y,
  x,
  maxsum = max(x) - min(x),
```

```
  weights = NULL,
  ss_within = NULL,
  edf = TRUE,
  parinit = NULL,
  last_delta = 1,
  ebic_xi = 0.3
)
```

Arguments

y	numeric vector with the response (missing values allowed).
x	numeric vector with the covariate (no missing values allowed).
maxsum	maximum sum of the γ_t, that is the warping budget M of the constraint $\sum_t \gamma_t \leq M$. It is expressed in the units of x and defaults to the range of x, the same upper end as the grid searched by auto_ssj_mle.
weights	optional numeric vector of positive observation weights, one per element of y. The observation variance becomes $H_t = \lambda\sigma^2/w_t$, so a point carrying weight w_t is treated as the mean of w_t independent observations. This is what makes it correct to collapse tied values of x to their mean and pass the number of replicates as the weight: the Gaussian likelihood factorises, so the weighted fit on the collapsed data is identical to the fit on the raw replicates. The default, NULL, means unit weights. Note a caveat specific to this function, which estimates the noise level rather than taking it as given. Collapsing replicates to their mean discards the within-group scatter, and that scatter is exactly what identifies the observation variance: on a test case with 150 observations at 40 distinct sites the raw fit recovered a variance of 0.021 against a true 0.0225, while the weighted collapsed fit returned 0.086. Pass the discarded scatter through ss_within to restore it; the estimates then agree with the raw ones to four or five significant digits. At a *fixed* smoothing parameter the discarded term does not involve f at all, which is why ssj_kfs, ssj_fix and ssj_atw reproduce the raw fit without it.
ss_within	optional pooled within-group sum of squares discarded when replicates were collapsed to their means, that is $\sum_i \sum_j (y_{ij} - \bar{y}_i)^2$. Supplying it, together with the replicate counts in weights, restores the term of the likelihood that the collapsing removed, so the estimates are those that the raw replicates would have produced. It has no effect on the fit at a fixed smoothing parameter; what it restores is the information identifying the observation variance, and hence lambda.
edf	logical: if TRUE (default) the effective degrees of freedom are computed as the trace of the hat matrix; if FALSE the non-zero γ_t are counted.
parinit	NULL or a numeric vector of length n+1 with starting values for the optimizer, namely $(\sigma, \lambda, \gamma_1, \ldots, \gamma_{n-1})$; pass the previous solution for warm starting.
last_delta	positive scalar: dummy spacing appended after the last observation (it does not affect the fit; default 1).
ebic_xi	numeric scalar in [0,1]: strength of the extended-BIC penalty for searching over the n-1 candidate jump locations (default 0.3; 0 reduces ebic to plain bic). The default is calibrated rather than taken from theory; see the note below.

Details

Since `lambda` enters only the observation variance $H_t = \lambda\sigma^2$ and never the transition matrix, its score does not need the extra correction that γ requires: the plain Koopman-Shephard formula applies directly, $\partial\ell/\partial\lambda = (\sigma^2/2)\sum_t(e_t^2 - d_t)$. Internally `lambda` is optimized via $\mu = \log\lambda$, which guarantees $\lambda > 0$ without a box constraint and avoids the scale instability observed when optimizing the raw `lambda` directly (its sensible range can span orders of magnitude, as for any variance ratio); the score is then a one-line chain-rule correction, $\partial\ell/\partial\mu = \lambda\,\partial\ell/\partial\lambda$. Both `parinit` and the returned `pars` are unaffected and stay in terms of the raw `lambda`.

When `parinit` is not supplied, the success of the optimizer depends on the path it traverses, not just on how close the starting `lambda` is to the eventual optimum. Rather than relying on a single formula, `ssj_mle` therefore tries a fixed grid of starting values for `lambda` spanning several orders of magnitude and keeps whichever converges to the highest likelihood.

Tied values of `x` are collapsed to their weighted mean before fitting, and the fitted values are returned at the vector of unique, sorted `x` available in the `x` slot of the output. This keeps the problem well posed: with ties the interval between two replicates has zero length, and a warp placed there would be a discontinuity between two observations recorded at the same `x`, which is not identified. The replicate counts become observation weights automatically, so no information is lost.

Value

An object of class `ssj` (a list, so that all the slots below remain directly accessible) with a `print` and a `plot` method. Its slots are:

- `opt`: the output of the `nloptr` optimizer.
- `nobs`: number of observations.
- `df`: model's degrees of freedom.
- `maxsum`: the `maxsum` argument.
- `loglik`: Gaussian log-likelihood at the optimum, on the scale of `y` as supplied. Note that the diffuse initial conditions are approximated by a large proper prior, so the absolute value carries an additive constant set by that convention; differences across `maxsum` for one data set, which is what the information criteria compare, are unaffected.
- `pars`: named vector (sigma, lambda); `lambda` is estimated.
- `gamma`: estimated time-warp vector of length `n-1`.
- `ic`: vector of information criteria (aic, aicc, bic, hq, ebic).
- `smoothed_level`: smoothed estimate of f at each `x`.
- `var_smoothed_level`: pointwise variance of the smoothed level.
- `dist_level`, `dist_slope`: smoothed level and slope disturbances (`NULL` when `edf = FALSE`).
- `std_dist_level`, `std_dist_slope`: the corresponding standardised (auxiliary residual) versions.
- `x`: the ordered `x` variable.

Choosing `ebic_xi`

The default is the value that minimises worst-case regret in root mean squared error over a factorial design of 48 cells – four sample sizes, three noise levels and four counts of discontinuities, 17,400 replications in all. The experiment is shipped as `inst/scripts/xi_calibration.R`. For this estimator the optimum is flat in n, at 0.27 to 0.33, so a constant is appropriate; `ssj_kfs` and `ssj_atw` were calibrated separately and carry different defaults, because they are different procedures even where they fit the same model.

The value sits below the Chen and Chen (2008) consistency floor of about 0.5 for $p = n - 1$ candidates, and that is not an oversight. Their derivation assumes the base penalty charges only for dimension, with the whole cost of searching the model space carried by the ξ term. Here part of it is already paid: `df` is the trace of the smoothing matrix and grows with each activated warp, so the plain BIC term levies about $0.16 \log n$ per declared jump before the extended term adds anything. Expressed on the ξ scale that implicit charge is roughly 0.08.

Detection accuracy cannot be used to calibrate ξ on this problem: with $n - 1$ candidates and a handful of true discontinuities the negative class dominates, and both balanced accuracy and the Matthews coefficient give optima that wander across the design. Larger values declare fewer jumps if that is what a particular application needs.

References

Pelagatti, M., Maranzano, P. and Gatto Ronchieri, G. C. (2026). Smoothing splines with some discontinuities. *arXiv preprint* arXiv:2610.33137. doi:10.48550/arXiv.2610.33137

Chen, J. and Chen, Z. (2008). Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, **95**(3), 759–771. doi:10.1093/biomet/asn034

Koopman, S. J. and Shephard, N. (1992). Exact score for time series models in state space form. *Biometrika*, **79**(4), 823–826. doi:10.1093/biomet/79.4.823

See Also

`auto_ssj_mle`, `ssj_kfs`, `ssj_atw`, `ssj_fix`.

Examples

```
# 'maxsum' is a warping budget in the units of x; the range of eruptions
# is about 3.5, so a budget of 1 is already generous.
# thinned for speed: the full 'faithful' data set works just as well
fa <- faithful[seq(1, nrow(faithful), by = 3), ]
plot(fa$eruptions, fa$waiting)
of_mle <- ssj_mle(y = fa$waiting, x = fa$eruptions, maxsum = 1)
lines(x = of_mle$x, y = of_mle$smoothed_level, col = "darkgreen", lwd = 2)
```

`trigseas` *Trigonometric seasonal variables*

Description

Produces a matrix of seasonal sinusoids to be used as regressors. By default, for $t = 1, \ldots, n$ and $j = 1, \ldots, \lfloor s/2 \rfloor$, it computes

$$\cos(2\pi jt/s), \qquad \sin(2\pi jt/s).$$

Notice that if s is even the sine function at the highest frequency is omitted, because it equals zero for all t. The user can select a different set of harmonics.

Usage

```
trigseas(n, s, harmonics = NULL)
```

Arguments

`n`	length of the time series.
`s`	seasonal period.
`harmonics`	vector of harmonics to be used: the cosine and sine functions are computed at frequencies `2*pi*harmonics/s`; if there is an element of `harmonics` for which `2*pi*harmonics/s` equals π, the corresponding sine is omitted.

Value

A matrix with `n` rows and as many columns as the harmonics (by default $s - 1$).

See Also

`dummyseas` for the seasonal-dummy alternative.

Examples

```
y <- log(AirPassengers)
X <- trigseas(length(y), 12)
X <- cbind(X, t = seq_along(y))
reg <- lm(y ~ X)
```

Index

∗ **datasets**
demand_temp, 19
employed_IT, 21
load_temp, 30

auto_hpfj, 3, *22*
auto_hpfj_fix, 4, *24*
auto_hpfjx, 5, *26*
auto_ssj_atw, 6, *8–12*, *17*, *41, 42*
auto_ssj_atw_cv, 8, *11, 12*
auto_ssj_atw_gcv, *8, 9*, 10
auto_ssj_cssd, 12, *46, 47*
auto_ssj_fix, 14, *49*
auto_ssj_kfs, 15, *17*, *50–52*
auto_ssj_mle, *16*, 17, *51*, *53*, *55*

BIC.hpj, 18

demand_temp, 19
dummyseas, 20, *26*, *56*

employed_IT, 21

hpfj, *3, 4*, *6*, 21, *24–27*
hpfj_fix, *4, 5*, *22*, 23, *27*
hpfjx, *5, 6*, 25, *27*
hpj, *4–6*, *21–25*, 26

llt, 28
load_temp, *19, 20*, 30
logLik.hpj, 31

mse, 31

nobs.hpj, 32

plot, *41*, *45*, *47*, *49*, *51*, *54*
plot.hpj, *27*, 33, *35*
plot.ssj, 34, *37*, *40*, *44*
predict.ssj, 36
predict_cssd_dp, *13*, *36*, 37, *37*, *47*
print, *41*, *45*, *47*, *49*, *51*, *54*
print.hpj, *27*, 38
print.ssj, *35*, 39, *44*

sample, *13*
set.seed, *13*
ssj_atw, *6–12*, 40, *43*, *48, 49*, *52, 53*, *55*
ssj_breaks, *35*, *37*, *39, 40*, 43
ssj_cssd, *37*, 44, *47*
ssj_cssd_dp, *13*, *37, 38*, *45*, 46
ssj_fix, *14, 15*, *40*, *42–44*, 47, *52, 53*, *55*
ssj_kfs, *15, 16*, *42, 43*, *48, 49*, 50, *52, 53*, *55*
ssj_mle, *10*, *17, 18*, *42, 43*, *48, 49*, *51*, 52, *52*

trigseas, *20*, *26*, 56