Package ‘SpatialFoFReg’

December 7, 2025

Type Package

Title Spatial Function-on-Function Regression Models

Version 1.0

Depends R (>= 3.5.0), fda, MASS, Matrix, matrixStats, methods, quantreg, ald

LazyLoad yes

ByteCompile TRUE

Encoding UTF-8

Maintainer Ufuk Beyaztas <ufukbeyaztas@gmail.com>

Description Functions for implementing methods for spatial function-on-function regression models.

License GPL-3

Contents

`predict_sffr2SLS` *Out–of–sample prediction for Penalised Spatial FoFR models*

Description

Given a fitted model returned by `sffr_pen2SLS`, this function produces predicted functional responses at new spatial units whose functional covariates and spatial weight matrix are supplied by the user. A fixed–point solver enforces the spatial autoregressive feedback implicit in the SFoFR model.

Usage

```
predict_sffr2SLS(object, xnew, Wnew)
```

Arguments

`object`	An object of class `"sffr2SLS"`: the list produced by `sffr_pen2SLS`. At minimum it must contain `gpx`, `gpy`, `K0`, `Ky`, `Kx`, and the B–spline coefficient matrices `b0_mat`, `b_mat`, `r_mat`.
`xnew`	Numeric matrix of dimension $n_{\text{new}} \times |\text{gpx}|$, holding the functional covariate for the *new* spatial units, evaluated on the same predictor grid used during model fitting.

Wnew Row–normalised $n_{\text{new}} \times n_{\text{new}}$ spatial weight matrix that captures proximity among the *new* units. Its definition should mirror that of the training matrix (e.g.\ inverse distance, k–nearest neighbours, etc.).

Details

Let $\widehat{\beta}_0(t)$, $\widehat{\beta}(t,s)$, and $\widehat{\rho}(t,u)$ be the estimated surfaces stored in `object`. For each new unit i the algorithm first forms the non–spatial regression prediction

$$\widehat{G}_i(t) \;=\; \widehat{\beta}_0(t) + \int_0^1 X_i(s)\,\widehat{\beta}(t,s)\,ds,$$

computed efficiently by pre-evaluated B–spline bases. Spatial feedback is then introduced by iterating

$$Y_i^{(\ell+1)}(t) \;=\; \widehat{G}_i(t) + \sum_{j=1}^{n_{\text{new}}} w_{ij} \int_0^1 Y_j^{(\ell)}(u)\,\widehat{\rho}(t,u)\,du,$$

until the sup–norm difference between successive curves falls below `1e-3` or `1,000` iterations are reached. Convergence is guaranteed when $\|\widehat{\rho}\|_\infty < 1/\|Wnew\|_\infty$, a condition typically satisfied by the fitted model if the training weight matrix met it during estimation.

Value

A numeric matrix of dimension $n_{\text{new}} \times |\text{gpy}|$ containing the predicted functional responses evaluated on `gpy`. Row i corresponds to the i-th row of `xnew`.

Note

If the new weight matrix induces very strong dependence, the fixed–point iterations may converge slowly. Consider scaling `Wnew` to have $\|Wnew\|_\infty \le 1$ or relaxing the tolerance.

Author(s)

Eylul Fidan, Ufuk Beyaztas, and Soutir Bandyopadhyay

See Also

`sff_dgp` for simulated data generation; `sffr_pen2SLS` for model fitting.

Examples

```
## Not run: -------------------------------------------------------------------
## 1. Fit a model on small simulated data
# train <- sff_dgp(n = 500, rf = 0.5)
# lam   <- list(lb = c(10^{-3}, 10^{-2}, 10^{-1}),
#                       lrho = c(10^{-3}, 10^{-2}, 10^{-1}))

# fit <- sffr_pen2SLS(train$Y, train$X, train$W,
#                     gpy = seq(0, 1, length = 101),
#                     gpx = seq(0, 1, length = 101),
#                     K0 = 10, Ky = 10, Kx = 10,
#                     lam_cands = lam)

## 2. Simulate NEW covariates and a compatible weight matrix
```

```
# test <- sff_dgp(n = 1000, rf = 0.5)  ## we keep only X and W
# pred <- predict_sffr2SLS(fit, xnew = test$X, Wnew = test$W)
## End(Not run)
```

---

predict_sff_qr *Prediction for Spatial Function-on-Function Quantile IV Model*

Description

Generates fitted values from a previously estimated spatial function-on-function quantile instrumental-variable model (via `sff_qr`) for new functional predictor data.

Usage

```
predict_sff_qr(object, xnew, Wnew)
```

Arguments

`object`	An object returned by `sff_qr` containing the estimated surfaces `b0hat`, `bhat`, `rhohat` and the grids `grid_x`, `grid_y`.
`xnew`	A numeric $n_{\text{new}} \times p_x$ matrix of functional predictor observations at grid points `grid_x`.
`Wnew`	A numeric $n_{\text{new}} \times n_{\text{new}}$ spatial weight matrix (row-normalised) for the new observations, used for the spatial feedback component.

Details

This function computes predicted values $\widehat{Y}_i(t_m)$ for new data as follows:

1. It uses the estimated intercept surface $\widehat{\beta}_0(t)$, the estimated coefficient surface $\widehat{\beta}(t,s)$ and the new predictor matrix X_{new} to compute the initial fitted surface

$$\widehat{Y}_i^{(0)}(t_m) = \widehat{\beta}_0(t_m) + \int_{s_\ell} \widehat{\beta}(t_m, s_\ell)\; X_{i,\text{new}}(s_\ell)\; \Delta s.$$

2. It then incorporates the spatial feedback component using the estimated feedback surface $\widehat{\rho}(u,t)$ and the matrix W_{new}:

$$\widehat{Y}_i^{(k+1)}(t_m) = \widehat{\beta}_0(t_m) + \int_{s_\ell} \widehat{\beta}(t_m, s_\ell)\; X_{i,\text{new}}(s_\ell)\; \Delta s + \sum_j W_{\text{new},ij} \int_{t_{m'}} \widehat{Y}_j^{(k)}(t_{m'})\; \widehat{\rho}(t_{m'}, t_m)\; \Delta t.$$

The algorithm iterates until convergence (maximum absolute change \(<0.001\)) or until a maximum number of iterations (1000) is reached.

Value

A numeric matrix of dimension $n_{\text{new}} \times p_y$ giving predicted functional responses $\widehat{Y}_i(t_m)$ on the grid `grid_y`.

Author(s)

Eylul Fidan, Ufuk Beyaztas & Soutir Bandyopadhyay

Examples

```
## Not run: ------------------------------------------------------------
## 1. Fit a model on small simulated data
# train <- sff_dgp(n = 500, rf = 0.5)
# fit <- sff_qr(
#   y   = train$Y,
#   x   = train$X,
#   W   = train$W,
#   gpy = seq(0, 1, length.out = 101),
#   gpx = seq(0, 1, length.out = 101)
# )
#
## 2. Simulate NEW covariates and a compatible weight matrix
# test <- sff_dgp(n = 1000, rf = 0.5)  ## we keep only X and W
# pred <- predict_sff_qr(fit, xnew = test$X, Wnew = test$W)
## End(Not run)
```

`sffr_pen2SLS` *Penalised Spatial Two–Stage Least Squares for SFoFR*

Description

Fits the penalised spatial function-on-function regression (SFoFR) model via the two–stage least–squares (Pen2SLS) estimator introduced by Beyaztas, Shang and Sezer (2025). It selects optimal smoothing parameters, estimates regression and spatial autocorrelation surfaces, and (optionally) builds percentile bootstrap confidence bands.

Usage

```
sffr_pen2SLS(
  y, x, W, gpy, gpx,
  K0, Ky, Kx,
  lam_cands,
  boot       = FALSE,
  nboot      = NULL,
  percentile = NULL
)
```

Arguments

y	n x `length(gpy)` matrix of functional responses evaluated on grid `gpy`.
x	n x `length(gpx)` matrix of functional predictors evaluated on grid `gpx`.
W	n x n row–normalised spatial weight matrix, typically inverse–distance.
gpy	Numeric vector of response evaluation points $t \in [0, 1]$.
gpx	Numeric vector of predictor evaluation points $s \in [0, 1]$.
K0	Integer; number of basis functions for the intercept $\beta_0(t)$.
Ky	Integer; number of basis functions in the *response* direction of the bivariate surfaces $\rho(t, u)$ and $\beta(t, s)$.
Kx	Integer; number of basis functions in the *predictor* direction of the regression surface $\beta(t, s)$.

`lam_cands`	Two–column matrix or data frame whose rows contain candidate smoothing pairs $(\lambda_\rho,\ \lambda_\beta)$ to be ranked by BIC.
`boot`	Logical; if `TRUE` percentile bootstrap confidence intervals are produced.
`nboot`	Number of bootstrap resamples. Required when `boot = TRUE`.
`percentile`	Desired CI nominal width in percent (e.g., `95`). Required when `boot = TRUE`.

Details

The estimator minimises the penalised objective

$$\|Z^*\{\mathrm{vec}(Y) - \Pi\theta\}\|^2 \ + \ \frac{1}{2}\lambda_\rho P(\rho) \ + \ \frac{1}{2}\lambda_\beta P(\beta),$$

where $\theta = (\mathrm{vec}\,\rho, \mathrm{vec}\,\beta)$ are tensor-product B-spline coefficients, Z^* is the projection onto instrumental variables, and $P(\cdot)$ are Kronecker-sum quadratic roughness penalties in both surface directions. Candidate smoothing pairs are scored by the Bayesian Information Criterion

$$\mathrm{BIC} = -2\log\mathcal{L} + \omega\log n,$$

with log-likelihood based on squared residuals and ω equal to the effective degrees of freedom.

If `boot = TRUE`, residuals are centred, resampled, and the entire estimation procedure is repeated `nboot` times. Lower and upper percentile bounds are then extracted for $\beta(t,s)$, $\rho(t,u)$, and $\widehat{Y}_i(t)$.

Value

A named `list`:

b0hat Estimated intercept curve $\widehat{\beta}_0(t)$.

bhat Matrix of $\widehat{\beta}(t,s)$ values.

rhohat Matrix of $\widehat{\rho}(t,u)$ values.

b0_mat, b_mat, r_mat Raw coefficient matrices of B-spline basis weights for β_0, β, and ρ.

fitted.values $\widehat{Y}_i(t)$ matrix.

residuals $\widehat{\varepsilon}_i(t)$ matrix.

CI_bhat Two-element list with lower/upper percentile surfaces (NULL unless `boot = TRUE`).

CI_rhohat Analogous list for ρ.

CIy Percentile bands for the fitted responses.

gpy, gpx, K0, Ky, Kx Returned for convenience.

Author(s)

Eylul Fidan, Ufuk Beyaztas, and Soutir Bandyopadhyay

Examples

```
## Not run: ---------------------------------------------------------------
## 1. simulate data
# sim <- sff_dgp(n = 100, rf = 0.7)
## 2. candidate smoothing grid (four pairs)
# lam <- list(lb = c(10^{-3}, 10^{-2}, 10^{-1}),
#                       lrho = c(10^{-3}, 10^{-2}, 10^{-1}))
## 3. fit model without bootstrap
# fit <- sffr_pen2SLS(
#   y         = sim$Y,
#   x         = sim$X,
#   W         = sim$W,
#   gpy       = seq(0, 1, length.out = 101),
#   gpx       = seq(0, 1, length.out = 101),
#   K0        = 10,
#   Ky        = 10,
#   Kx        = 10,
#  lam_cands  = lam,
#  boot       = FALSE
# )
## End(Not run)
```

`sff_dgp` *Simulate data from a Spatial Function-on-Function Regression model under multiple error structures*

Description

Generates synthetic functional predictors and responses from the spatial function-on-function regression (SFoFR) data-generating process. The generator supports three distinct error structures, homoscedastic Gaussian, signal-dependent heteroscedastic Gaussian with upper-tail contamination, and asymmetric Laplace-to enable flexible simulation scenarios.

Usage

```
sff_dgp(
  n,
  nphi     = 10,
  gpy      = NULL,
  gpx      = NULL,
  rf       = 0.9,
  sd.error = 0.01,
  tol      = 0.001,
  max_iter = 1000,
  case     = c("1","2","3")
)
```

Arguments

`n`	Number of spatial units (curves) to generate.
`nphi`	Number of sine *and* cosine basis functions used to build each functional predictor. Total latent scores generated are therefore `2 * nphi`.

`gpy` Numeric vector of evaluation points for the response domain $t \in [0, 1]$. Defaults to an equally–spaced grid of 101 points.

`gpx` Numeric vector of evaluation points for the predictor domain $s \in [0, 1]$. Defaults to an equally–spaced grid of 101 points.

`rf` Scalar in $(0, 1)$ controlling the strength of spatial autocorrelation through the surface $\rho(t, u)$. Values closer to 1 yield stronger dependence.

`sd.error` Scale parameter controlling the magnitude of the noise component. Its interpretation depends on the chosen case.

`tol` Absolute tolerance used in the fixed–point iteration that solves the spatial autoregressive operator equation (stopping rule on the sup–norm of successive iterates).

`max_iter` Maximum number of fixed–point iterations. Prevents infinite looping when strong spatial feedback and small `tol` interact.

`case` Specifies the noise generation mechanism:

"1" Homoscedastic Gaussian errors: $\varepsilon_i(t) \sim \mathcal{N}(0, \sigma^2)$ with constant variance.

"2" Signal-dependent heteroscedastic Gaussian errors with upper-tail contamination. The errors are generated as $\varepsilon_i(t) = \sigma_i(t)\eta_i(t) + B_i(t)c_{\text{out}}\sigma_i(t)$, where $\eta_i(t) \overset{i.i.d.}{\sim} \mathcal{N}(0, 1)$ and $\sigma_i(t)$ is a time-location specific scale that increases smoothly with the local signal magnitude. Letting $m_i(t)$ denote the absolute signal component (e.g., $m_i(t) = |\mu_i(t)|$ with $\mu_i(t)$ the non-error part of $Y_i(t)$), the code first rescales $m_i(t)$ to $[0, 1]$ via

$$m_i^\star(t) = \frac{m_i(t) - \min_{j,u} m_j(u)}{\max_{j,u} m_j(u) - \min_{j,u} m_j(u) + 10^{-8}},$$

and sets $\sigma_i(t) = \sigma_\varepsilon\{1 + a_\sigma m_i^\star(t)\}$, so that $\sigma_i(t)$ ranges roughly from σ_ε to $(1 + a_\sigma)\sigma_\varepsilon$. Upper-tail contamination is introduced through a Bernoulli indicator $B_i(t) \sim \text{Bernoulli}\{p_i(t)\}$, with $p_i(t) = p_0 + p_1 m_i^\star(t)$ (capped at a maximum probability, e.g. $p_i(t) \leq 0.3$), and a positive shift of size $c_{\text{out}}\sigma_i(t)$ whenever $B_i(t) = 1$. This yields a conditional error distribution that is both heteroscedastic and strongly right–skewed in high–signal regions.

"3" Asymmetric Laplace errors: $\varepsilon_i(t) \sim \text{ALD}(0, \sigma, p = 0.5)$, introducing heavy tails and asymmetry.

Details

The generator mimics the penalised SFoFR set-up:

$$Y_i(t) = \sum_{j=1}^{n} w_{ij} \int_0^1 Y_j(u)\rho(t, u)du + \int_0^1 X_i(s)\beta(t, s)ds + \varepsilon_i(t),$$

where

- w_{ij} are row-normalised inverse–distance weights,
- $X_i(s)$ is built from Fourier scores $\xi_{ijk} \sim \mathcal{N}(0, 1)$ and damped basis functions $\phi_k^{\cos}(s) = (k^{-3/2})\sqrt{2}\cos(k\pi s)$ and $\phi_k^{\sin}(s) = (k^{-3/2})\sqrt{2}\sin(k\pi s)$,
- the regression surface is $\beta(t, s) = 2 + s + t + 0.5\sin(2\pi st)$,
- the spatial autocorrelation surface is $\rho(t, u) = rf\,(1 + ut)/(1 + |u - t|)$,

- the error structure is determined by `case`, as described above.

Given the contraction condition $\|\rho\|_\infty < 1/\|W\|_\infty$, the Neumann series defining $(\mathbb{I} - \mathcal{T})^{-1}$ converges and the solution is obtained by simple fixed-point iterations until the change is below `tol`.

Value

A named `list` with components

Y `n x length(gpy)` matrix of observed functional responses on the grid gpy.

Y_true Same dimension as `Y`; noise–free latent responses before adding $\varepsilon_i(t)$.

X `n x length(gpx)` matrix of functional predictors.

W `n x n` row-normalised spatial weight matrix based on inverse distances.

rho `length(gpy) x length(gpy)` matrix containing $\rho(t, u)$ evaluated on the response grid.

beta `length(gpx) x length(gpy)` matrix containing $\beta(t, s)$ evaluated on the Cartesian product of the predictor and response grids.

Author(s)

Eylul Fidan, Ufuk Beyaztas, and Soutir Bandyopadhyay

Examples

```
## Not run: ---------------------------------------------------------------
## generate datasets under three scenarios
# dat1 <- sff_dgp(n = 100, rf = 0.6, case = "1")  # Homoscedastic Gaussian
# dat2 <- sff_dgp(n = 100, rf = 0.6, case = "2")  # Heteroscedastic + upper-tail contamination
# dat3 <- sff_dgp(n = 100, rf = 0.6, case = "3")  # Asymmetric Laplace
## End(Not run)
```

`sff_qr` *Spatial Function-on-Function Quantile Instrumental-Variable Estimation*

Description

Performs penalised two-stage spatial function-on-function quantile regression with tensor-product B-splines and smoothing-parameter selection.

Usage

```
sff_qr(
  y, x, W,
  gpx, gpy,
  K0 = 10, Kx = 10, Ky = 10,
  tau = 0.5,
  lam_cands = NULL,
  lb = 0.01,
  lrho = 0.01,
  alpha = 1e-2,
  ridge_eps = 1e-8,
```

```
  osqp_opts = list(
  trace  = 0,
  maxit = 8000,
  factr  = 1e7),
  verbose = TRUE,
  BIC = FALSE
)
```

Arguments

y	A numeric $n \times p_y$ matrix of functional responses $Y_i(t_m)$, with n observations and p_y grid-points at times given by gpy.
x	A numeric $n \times p_x$ matrix of functional predictors $X_i(s_\ell)$, with n observations and p_x grid-points at times given by gpx.
W	A numeric $n \times n$ spatial weight matrix (row-normalised) capturing spatial autoregression of the functional response.
gpx	A numeric vector of length p_x giving the grid points s_ℓ at which x is observed.
gpy	A numeric vector of length p_y giving the grid points t_m at which y is observed.
K0	Integer: number of basis functions for the intercept surface $\beta_0(t)$. Default is 10.
Kx	Integer: number of basis functions in the s-direction for the tensor-product basis for $\beta(t, s)$. Default is 10.
Ky	Integer: number of basis functions in the t-direction for the tensor-product basis for $\beta(t, s)$ and for $\rho(t, u)$. Default is 10.
tau	Quantile level $\tau \in (0, 1)$ for the quantile regression. Default is 0.5 (median).
lam_cands	A list with components lb and lrho giving candidate values of smoothing parameters λ_β and λ_ρ. If non-NULL then smoothing-parameter selection via BIC is activated.
lb	Smoothing parameter λ_β for the roughness penalty on $\beta(t, s)$. Ignored if lam_cands is provided.
lrho	Smoothing parameter λ_ρ for the roughness penalty on $\rho(t, u)$. Ignored if lam_cands is provided.
alpha	Smoothing parameter for the smoothed pinball ("Moreau/Huberised") loss approximation. Smaller α closer to exact check-loss. Default is 1e-2.
ridge_eps	Small ridge parameter added to the penalty matrix to ensure positive-definiteness / numeric stability. Default is 1e-8.
osqp_opts	List of control parameters passed to the optim solver (using the "L-BFGS-B" method) in the second stage. Key components include: maxit (maximum number of iterations), factr (convergence tolerance control), and trace (verbosity level)
verbose	Logical. If TRUE, progress messages are printed from Stage 1 and Stage 2 of the algorithm.
BIC	Logical. If TRUE and lam_cands is non-NULL, the smoothing parameters are selected via the BIC criterion; if FALSE, the user-supplied lb and lrho are used directly.

Details

This function implements the two-stage instrumental-variable quantile regression for spatial function-on-function models. In Stage 1, a quantile regression of the spatially-lagged functional response $W\,Y_i(t_m)$ on basis-expanded instruments (including $(X_i(s),\,W\,X_i(s),\,W^2\,X_i(s))$) is performed for each grid-point combination via a call to `rq`. This produces fitted values $\widehat{W\,Y}_i(t_m)$. In Stage 2, the design matrix is constructed by combining the basis expansions for the intercept surface, the tensor-product basis for $\beta(t,s)$, and a basis representation of the spatial feedback surface $\rho(t,u)$. A penalised smoothed-quantile objective is then solved

$$\min_{\beta,\rho}\sum_{i,m}\eta_\tau\big(Y_i(t_m)-\widehat{Y}_i(t_m)\big)\;+\;\tfrac{1}{2}\,\beta^\top P\,\beta,$$

where $\eta_\tau(u)$ is the check-loss (approximated by a smoothed version) and P is the block-diagonal penalty matrix with roughness penalties controlled by λ_β and λ_ρ. If `lam_cands` is provided, a grid-search over $(\lambda_\beta,\lambda_\rho)$ is performed and the optimal pair minimises the BIC criterion:

$$\mathrm{BIC}(\lambda_\rho,\lambda_\beta)=\log\Big(\tfrac{1}{nM}\sum_{i,m}\eta_\tau(Y_i(t_m)-\widehat{Y}_i(t_m))\Big)\;+\;\frac{\log(nM)\,\mathrm{df}(\lambda_\rho,\lambda_\beta)}{nM},$$

where n is the number of individuals, M the number of grid-points, and $\mathrm{df}(\cdot)$ denotes the effective degrees of freedom (approximated here by the number of free basis coefficients). The selected $\widehat{\lambda}_\beta$ and $\widehat{\lambda}_\rho$ are then used for the final fit reported in the surfaces component of the result.

Value

A list with the following components:

`surfaces` A list containing:

- `grid_x` — the vector of grid locations s_ℓ (same as input gpx),
- `grid_y` — the vector of grid locations t_m (same as input gpy),
- `b0hat` — the estimated intercept surface $\widehat{\beta}_0(t)$, as a numeric vector of length p_y,
- `bhat` — the estimated coefficient surface $\widehat{\beta}(t,s)$, as a numeric matrix of dimension $p_y \times p_x$,
- `rhohat` — the estimated spatial-feedback surface $\widehat{\rho}(t,u)$, as a numeric matrix of dimension $p_y \times p_y$.

`fitted` A numeric $n \times p_y$ matrix of fitted values $\widehat{Y}_i(t_m)$.

`resid` A numeric $n \times p_y$ matrix of residuals $Y_i(t_m) - \widehat{Y}_i(t_m)$.

Author(s)

Eylul Fidan, Ufuk Beyaztas & Soutir Bandyopadhyay

Examples

```
## Not run: ----------------------------------------------------------------
## simulate data
# sim <- sff_dgp(n = 250, rf = 0.7)
## fit model
# fit <- sff_qr(
#   y          = sim$Y,
#   x          = sim$X,
```

```
#   W          = sim$W,
#   gpy        = seq(0, 1, length.out = 101),
#   gpx        = seq(0, 1, length.out = 101)
# )
## End(Not run)
```