Package: BsplineQuantReg (via r-universe)

June 23, 2026

Type Package

Title 'Constrained Quantile Regression with Cubic B-Splines'

Version 0.1.0

Date 2026-06-01

Description Quantile regression with cubic B-splines under monotonicity and convexity constraints using the Karlin-Studden SOCP formulation. The method is described in Abbes (2026) <doi:10.5281/zenodo.17427913>. This R implementation is intended for demonstration and prototyping; all B-spline and polynomial functions have been rewritten for consistency. A faster version written in 'Python' is available at <https://github.com/alexandreabbes/Constrained-Quantile-Regression-with-cubic-splines>.

License GPL-3

URL https://github.com/alexandreabbes/BsplineQuantReg, https://doi.org/10.5281/zenodo.17427913

BugReports https://github.com/alexandreabbes/BsplineQuantReg/issues

Depends R (>= 3.5.0)

Imports CVXR

Suggests cobs,quantreg,pracma

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Alexandre Abbes [aut, cre]

Maintainer Alexandre Abbes <alexandre.abbes@proton.me>

Config/pak/sysreqs cmake pkg-config

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

Date/Publication 2026-06-23 15:28:18 UTC

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

RemoteRef HEAD

RemoteSha 493e11b6a5468019d1332fdb114e2ba7584e15ca

Contents

`apply_karlin_constraints`

Karlin-Studden constraints for positivity

Description

Applies Karlin-Studden SOCP constraints to ensure positivity of a quadratic polynomial on the interval [0,1].

Usage

```
apply_karlin_constraints(p2, p1, p0, z0, verbose = FALSE)
```

Arguments

`p2`	Coefficient of u^2
`p1`	Coefficient of u
`p0`	Constant term
`z0`	Auxiliary SOCP variable
`verbose`	boolean FALSE (default) or TRUE.

Value

List of CVXR constraints

`bs_direct` *Direct evaluation of a B-spline basis*

Description

Computes the values of all B-spline basis functions at given points.

Usage

```
bs_direct(Basis, xvalues)
```

Arguments

`Basis`	Object returned by `Bspline_base`
`xvalues`	Vector of evaluation points

Value

Matrix of basis function values (n_splines x length(xvalues))

`Bspline_base` *Build B-spline basis in piecewise polynomial form Computes local polynomial coefficients for each B-spline basis function on each interval. Polynomials are expressed in the canonical basis Uses De Boor's recursion formula.*

Description

Build B-spline basis in piecewise polynomial form Computes local polynomial coefficients for each B-spline basis function on each interval. Polynomials are expressed in the canonical basis Uses De Boor's recursion formula.

Usage

```
Bspline_base(sn, degree = 3, der = 0, verbose = FALSE)
```

Arguments

`sn`	Extended knot vector (including endpoint repetitions) This means if t0..tkn it the set of knots then sn should be given as a vector with "degree" times t_0 and t_kn at the begining and the ends. its length is number of intervals+1+2*degree.
`degree`	B-spline degree (default = 3 for cubic)
`der`	Derivative order (0 = original basis)
`verbose`	boolean FALSE (default) or TRUE.

Value

A list containing:

base	Coefficients in the local bases, in the form of an 3-d array [j,nu,coeff], j : the number of the spline in the basis, nu: the number of the interval in the extended notation, coeff : the coefficients in decreasing order convention on the local bases (t-s_nu)^l, l=3..1 base[j,,] is a matrix of piecewise polynomial function compatible with the pp-form.
base0	Coefficients in canonical basis (centered at 0) (1, t, t^2, t^3) centered at the interval origin.
knots	Extended knot vector
int_knots	Internal knots (effective partition including ends)
degree	Spline degree
n_splines	Number of basis functions
deriv_order	Applied derivative order

Examples

```
sn <- c(0,0,0,0,1,2,3,4,5,5,5,5)
basis <- Bspline_base(sn, degree=3)
x=(0:(5*100))/100
y=bs_direct(basis,x)
matplot(x,t(y))
#or simple:
#view_basis(basis)
```

Bspline_deriv *Differentiate a B-spline basis*

Description

Computes the basis of order der derivatives of a B-spline basis.

Usage

```
Bspline_deriv(bspline, der = 2, verbose = FALSE)
```

Arguments

bspline	Object returned by Bspline_base
der	Derivative order
verbose	boolean FALSE (default) or TRUE.

Value

A list similar to Bspline_base for the derivative basis

bspline_to_deriv_coeffs_pp

Convert B-spline to derivative coefficients

Description

Converts a B-spline basis to normalized first and second derivative coefficients on each interval.

Usage

```
bspline_to_deriv_coeffs_pp(tn, degree = 3, xvalues = 0, verbose = FALSE)
```

Arguments

tn	Knot vector (effective partition, not extended)
degree	Spline degree (default = 3)
xvalues	Evaluation points for design matrix (0 = no evaluation)
verbose	boolean FALSE (default) or TRUE.

Value

A list containing:

d0	Design matrix (if xvalues provided)
d1	First derivative coefficients [a3, a2, a1] for each interval
d2	Second derivative values at knots

change_polynomial_base_taylor

Change polynomial basis (Taylor expansion)

Description

Converts a polynomial expressed in the basis (t-a)^k to its representation in the basis (t-b)^k using Taylor's formula. P(t) = sum c_k (t-a)^k P(t) = sum c"_k (t-b)^k

Usage

```
change_polynomial_base_taylor(coeffs_a, a, b)
```

Arguments

coeffs_a	Coefficients in basis centered at a (decreasing powers)
a	Original expansion point
b	New expansion point

Value

Coefficients in basis centered at b

---

is_beta *Check if package is beta version*

Description

Check if package is beta version

Usage

```
is_beta()
```

Value

TRUE if beta version

---

package_version *Get package version*

Description

Get package version

Usage

```
package_version()
```

Value

Current version string

poly_eval *Evaluate polynomial*

Description

Evaluates a polynomial at one or more points.

Usage

```
poly_eval(p, xvalues)
```

Arguments

p	Coefficient vector (decreasing powers)
xvalues	vector at which to evaluate the polynomial

Value

Vector of same length as xvalues : polynomial values at the points xvalues

Examples

```
# P(x) = 1 + x + x^2
poly_eval(c(1, 1, 1), c(0, 1, 2)) # returns c(1, 3, 7)
```

polyadd *Polynomial addition*

Description

Adds two polynomials represented by coefficients in decreasing power order.

Usage

```
polyadd(p1, p2, verbose = FALSE)
```

Arguments

p1	First polynomial (coefficient vector)
p2	Second polynomial (coefficient vector)
verbose	boolean FALSE (default) or TRUE.

Value

Coefficient vector of the sum

Examples

```
polyadd(c(1, 1), c(1, -1)) # returns c(2, 0)
```

`polyderiv` *Polynomial derivative Computes the derivative of order* `der` *of a polynomial.*

Description

Polynomial derivative Computes the derivative of order `der` of a polynomial.

Usage

```
polyderiv(p, der = 1)
```

Arguments

`p`	Coefficient vector (decreasing powers)
`der`	Derivative order (default = 1)

Value

Coefficients of the derivative polynomial

Examples

```
# P(x) = x^2 -> P'(x) = 2x
polyderiv(c(1, 0, 0), 1) # returns c(2, 0)
```

`polymul` *Polynomial multiplication*

Description

Multiplies two polynomials represented by coefficients in decreasing power order.

Usage

```
polymul(p1, p2, ord = 0, verbose = FALSE)
```

Arguments

`p1`	First polynomial (coefficient vector, decreasing powers)
`p2`	Second polynomial (coefficient vector, decreasing powers)
`ord`	Unused (compatibility parameter)
`verbose`	boolean FALSE (default) or TRUE.

Value

Coefficient vector of the product polynomial (decreasing powers)

Examples

```
# (1 + x) * (1 + x) = 1 + 2x + x^2
polymul(c(1, 1), c(1, 1)) # returns c(1, 2, 1)
```

`reduce_pol` *Reduce polynomial*

Description

Removes leading zeros from a polynomial coefficient vector.

Usage

```
reduce_pol(p, verbose = FALSE)
```

Arguments

`p`	Polynomial coefficient vector(coef in decreasing order)
`verbose`	boolean FALSE (default) or TRUE.

Value

Reduced vector (without leading zeros)

Examples

```
reduce_pol(c(0,0, 1, 1))
```

`Spline_der_knots` *Derivatives at knots of a B-spline*

Description

Computes derivative values of a B-spline at knots (efficient because it directly uses polynomial coefficients).

Usage

```
Spline_der_knots(Bspline, der = 1)
```

Arguments

Bspline	Object returned by Bspline_base
der	Derivative order (default = 1)

Value

Matrix of derivative values (n_splines x n_knots)

---

spline_eval *Evaluate a B-spline*

Description

Evaluates a spline (linear combination of B-splines) at given points.

Usage

```
spline_eval(Bspline, xvalues)
```

Arguments

Bspline	Spline object (list with coefficients on the Bspline basis, degree, extended knots)
xvalues	Vector of evaluation points

Value

vector of same length as xvalues, with Spline values at the requested points

Examples

```
{ # Create and evaluate a spline
sn <- c(0,0,0,0,1,2,3,4,5,5,5,5)
basis <- Bspline_base(sn, degree=3)
basis$coefficients <- runif(basis$n_splines)
y <- spline_eval(basis, seq(0,5,length=100))}
```

SplineConstQuantRegBs3 *Constrained quantile regression with cubic splines*

Description

Performs quantile regression using cubic B-splines, with optional monotonicity constraints (via Karlin-Studden) and convexity constraints.

Usage

```
SplineConstQuantRegBs3(
  xtab,
  ytab,
  knots,
  tau,
  monot = 0,
  convcons = 0,
  solver = "CLARABEL",
  weight = NULL,
  verbose = FALSE
)
```

Arguments

xtab	Predictor vector (x)
ytab	Response vector (y)
knots	Knot vector or number of knots (quantiles are then used)
tau	Quantile (between 0 and 1)
monot	Monotonicity constraint vector per interval: 1 = increasing, -1 = decreasing, 0 = unconstrained. If scalar, repeated.
convcons	Convexity constraint vector per knot: 1 = convex, -1 = concave, 0 = unconstrained. If scalar, repeated.
solver	CVXR solver to use (default = "CLARABEL")
weight	Observation weights (default = 1 for all)
verbose	boolean FALSE (default) or TRUE.

Value

A list containing:

coefficients	B-spline coefficients (including y mean)
degree	Spline degree (always 3)
knots	Knot vector used
int_knots	Same as knots (compatibility)

References

- Abbes, A. (2025). *Quantile regression with cubic polynomial splines under shape constraints with applications* . Zenodo. doi:10.5281/zenodo.16999784
- de Boor, C. (1978). *A Practical Guide to Splines*. Springer-Verlag. doi:10.1007/97814612-63333
- Karlin, S., & Studden, W. J. (1966). *Tchebycheff Systems: With Applications in Analysis and Statistics*. Interscience.
- Koenker, R., & Bassett, G. (1978). Regression Quantiles. *Econometrica*, 46(1), 33-50. doi:10.2307/1913643
- Koenker, R. (2025). quantreg: Quantile Regression. R package version 5.99. https://CRAN.R-project.org/package=quantreg
- Ng, P., & Maechler, M. (2024). cobs: Constrained B-Splines. R package version 1.3-8. https://CRAN.R-project.org/package=cobs

See Also

Related R packages:

- `quantreg` - Quantile regression with linear programming
- `cobs` - Constrained B-sines (linear and quadratic only)

Other implementations:

- MATLAB/Python versions: https://github.com/alexandreabbes/Constrained-Quantile-Regression-with-c

Examples

```
#optional set.seed(42)
x <- seq(0, 1, length=100)
y <- 2*x + sin(6*pi*x)/2 + rnorm(100, 0, 0.05)
knots <- quantile(x, probs=seq(0,1,length.out=10))

# Median quantile regression without constraints
fit <- SplineConstQuantRegBs3(x, y, knots, tau=0.5)

# With increasing monotonicity constraint
fit_monot <- SplineConstQuantRegBs3(x, y, knots, tau=0.5, monot=1)

# With convexity constraint
fit_convex <- SplineConstQuantRegBs3(x, y, knots, tau=0.5, convcons=1)
```

test_karlin_simple *Comprehensive test function*

Description

Runs quantile regression tests with and without constraints, and displays results. Demo function.

Usage

```
test_karlin_simple(verbose = FALSE, seed = NULL)
```

Arguments

verbose	boolean FALSE (default) or TRUE.
seed	(default=NULL) value for the random generator.

Value

No return value, produces plots.

Examples

```
test_karlin_simple()
```

view_basis *Visualize a B-spline functions basis*

Description

Plots all functions of a B-spline basis.

Usage

```
view_basis(Bspline, xvalues = 0)
```

Arguments

Bspline	Object returned by Bspline_base
xvalues	Vector of evaluation points for plotting (by default 100 points are computed in the knot range)

Value

No return value, called for side effects (generates a plot)

Index