Package 'GAReg'

May 7, 2026

Type Package

Title Genetic Algorithms in Regression

Version 0.1.2

Description Provides a genetic algorithm framework for regression problems requiring discrete optimization over model spaces with unknown or varying dimension, where gradient-based methods and exhaustive enumeration are impractical. Uses a compact chromosome representation for tasks including spline knot placement and best-subset variable selection, with constraint-preserving crossover and mutation, exact uniform initialization under spacing constraints, steady-state replacement, and optional island-model parallelization from Lu, Lund, and Lee (2010, <doi:10.1214/09-AOAS289>). The computation is built on the 'GA' engine of Scrucca (2017, <doi:10.32614/RJ-2017-008>) and 'changepointGA' engine from Li and Lu (2024, <doi:10.48550/arXiv.2410.15571>). In challenging high-dimensional settings, 'GAReg' enables efficient search and delivers near-optimal solutions when alternative algorithms are not well-justified.

License Apache License (== 2.0)

RoxygenNote 7.3.3

Depends R (>= 4.3.0)

Imports stats, splines, utils, methods, changepointGA, GA

URL https://github.com/mli171/GAReg

BugReports https://github.com/mli171/GAReg/issues

Suggests MASS, knitr, rmarkdown

Encoding UTF-8

VignetteBuilder knitr

LazyData true

NeedsCompilation no

Author Mo Li [aut, cre],
QiQi Lu [aut],
Robert Lund [aut],
Xueheng Shi [aut]

Maintainer Mo Li <mo.li@louisiana.edu>

Repository CRAN

Date/Publication 2026-04-27 05:10:23 UTC

Contents

`cptgaControl` *Build Control List for* `cptga/cptgaisl`

Description

Convenience constructor for GA control parameters used by `changepointGA::cptga` and `changepointGA::cptgaisl`. It merges named overrides into engine-specific defaults (.cptga.default or .cptgaisl.default), with light validation.

Usage

```
cptgaControl(
  ...,
  .list = NULL,
  .persist = FALSE,
  .env = asNamespace("GAReg"),
  .validate = TRUE,
  engine = NULL
)
```

Arguments

`...`	Named overrides for control fields (e.g., `popSize`, `pcrossover`, `minDist`, `numIslands`).
`.list`	Optional named list of overrides (merged with `...`).
`.persist`	Logical; if `TRUE`, persist updated defaults back into the target environment (not usually recommended in user code).
`.env`	Environment where defaults live (defaults to `parent.frame()`).

`.validate`	Logical; validate values/ranges (default `TRUE`).
`engine`	Character; one of `"cptga"` or `"cptgaisl"` to select the default set and validation rules.

Details

Unknown names are rejected. When both `...` and `.list` are present, they are combined, with later entries overwriting earlier ones.

Value

A list of class `"cptgaControl"`.

See Also

gareg_knots, .cptga.default, .cptgaisl.default

`crossover_fixknots` *Crossover Operator (Fixed-m) with Feasibility-First Restarts*

Description

Produces a child chromosome from two fixed-m parents (same number of knots) by alternately sampling candidate knot locations from the parents and enforcing the spacing constraint `diff(child) > minDist`. If a conflict is encountered, the routine restarts the construction up to a small cap.

Usage

```
crossover_fixknots(mom, dad, prange = NULL, minDist, lmax, N)
```

Arguments

`mom`, `dad`	Integer vectors encoding parent chromosomes: first entry m (number of change-points), followed by m ordered knot locations.
`prange`	Unused placeholder (kept for compatibility with other GA operators). Default `NULL`.
`minDist`	Integer; minimum spacing between adjacent knots in the child.
`lmax`	Integer; chromosome length (number of rows in the population matrix).
`N`	Integer; series length. Used to place the sentinel `N+1` at position `m+2`.

Details

Let mom and dad be chromosomes of the form c(m, tau_1, ..., tau_m, ...). This operator:

1. Initializes an empty child of size m.
2. Picks the first knot at random from mom or dad.
3. For each subsequent position $i = 2, \ldots, m$, considers the pair (mom[i], dad[i]) and chooses the first value that maintains the spacing constraint relative to the previously chosen knot (> minDist); if both work, one is chosen at random.
4. If no feasible choice exists at some step, the construction restarts from the first position (up to a small cap governed internally by up_tol).

The result is written back as a full-length chromosome with the sentinel N+1 in position m+2, and zeros elsewhere.

Value

An integer vector of length lmax encoding the child chromosome: c(m, child_knots, N+1, 0, 0, ...).

See Also

crossover_fixknots, mutation_fixknots, selectTau_uniform_exact, Popinitial_fixknots, gareg_knots

Examples

```
N <- 120
lmax <- 30
minDist <- 5
m <- 3
mom <- c(m, c(20, 50, 90), rep(0, lmax - 1 - m))
mom[m + 2] <- N + 1
dad <- c(m, c(18, 55, 85), rep(0, lmax - 1 - m))
dad[m + 2] <- N + 1
child <- crossover_fixknots(mom, dad, minDist = minDist, lmax = lmax, N = N)
child
```

FDRCalc — *False Discovery Rate (FDR) and True Positive Rate (TPR) from index labels*

Description

Computes the False Discovery Rate (FDR) and True Positive Rate (TPR, a.k.a. recall) by comparing a set of *true* labels to a set of *predicted* labels. Labels are treated as positive integer indices in $\{1, \ldots, N\}$. Duplicates are ignored (unique indices are used).

Usage

```
FDRCalc(truelabel, predlabel, N)
```

Arguments

truelabel	Integer vector of ground-truth positive indices (values in 1..N).
predlabel	Integer vector of predicted positive indices (values in 1..N).
N	Integer scalar; size of the full index universe (total number of candidates).

Details

Let truelabel and predlabel be sets of indices. The function derives the confusion-matrix counts:

- tp = $|truelabel \cap predlabel|$
- fp = $|predlabel \setminus truelabel|$
- fn = $|truelabel \setminus predlabel|$
- tn = $N - tp - fp - fn$

and returns

$$FDR = fp/(fp + tp), \qquad TPR = tp/(tp + fn).$$

Inputs are coerced to integer and uniqued. A warning is emitted if any label is < 1, and an error is thrown if any label exceeds N. If tn < 0, a warning is issued indicating that N may not reflect the full universe.

Value

A named list with components:

- fdr False Discovery Rate, $fp/(fp + tp)$ (NaN if fp+tp == 0).
- tpr True Positive Rate (recall), $tp/(tp + fn)$ (NaN if tp+fn == 0).
- fp, fn, tp, tn Confusion-matrix counts.

Edge cases

If predlabel is empty, fdr is NaN and tpr is 0 (unless truelabel is also empty, in which case both fdr and tpr are NaN). If truelabel is empty and predlabel non-empty, tpr is NaN and fdr is 1.

Examples

```
# Simple example
N <- 10
true <- c(2, 4, 7)
pred <- c(4, 5, 7, 7) # duplicates are ignored
FDRCalc(true, pred, N)

# Empty predictions
FDRCalc(true, integer(0), N)
```

```
# All correct predictions
FDRCalc(true, true, N)
```

`fixknotsIC` *Information criterion for a fixed–knot spline regression*

Description

Computes an information criterion (BIC, AIC, or AICc) for a regression of `y` on a spline basis of `x` when the number of interior knots is fixed. This is designed to be used as a fitness/objective function inside a GA search where the chromosome encodes the indices of the interior knots.

Usage

```
fixknotsIC(
  knot_bin,
  plen = 0,
  y,
  x,
  x_unique,
  x_base = NULL,
  fixedknots,
  degree = 3L,
  type = c("ppolys", "ns", "bs"),
  intercept = TRUE,
  ic_method = "BIC"
)
```

Arguments

`knot_bin`	Integer vector (chromosome). Gene 1 stores m, the number of interior knots. Genes 2:(1+m) are indices into `x_unique` selecting the m interior knots, followed by a sentinel equal to `length(x_unique)+1`. Only genes strictly before the first occurrence of `length(x_unique)+1` are treated as interior indices; genes after the sentinel are ignored. Interior indices must be in `2:(length(x_unique)-1)`, finite, and non-duplicated.
`plen`	Unused placeholder kept for API compatibility with other objective functions. Ignored.
`y`	Numeric response vector of length n.
`x`	Numeric predictor (same length as `y`) on which the spline basis is built.
`x_unique`	Optional numeric vector of unique candidate knot locations. If `NULL` or missing, it defaults to `sort(unique(x))`. Must contain at least $m + 2$ values (interior + two boundaries).
`x_base`	Optional matrix (or vector) of additional covariates to include linearly alongside the spline basis. If supplied, it is coerced to a matrix and column-bound to the design.

fixedknots	Integer m: the number of interior knots to use. Internally this determines how many indices are read from knot_bin.
degree	Integer polynomial degree for type="ppolys" and type="bs" (default 3L). Ignored for type="ns" (cubic).
type	One of c("ppolys","ns","bs"); forwarded to [splineX()].
intercept	Logical; forwarded to [splineX()]. For $m > 0$, the spline block is splineX(..., intercept=intercept) and no explicit 1-column is added here. If you add your own intercept in X, call splineX(..., intercept=FALSE).
ic_method	Character; which information criterion to return: "BIC", "AIC", or "AICc".

Details

We decode the interior indices up to the sentinel length(x_unique)+1, validate them (finite, interior, non-duplicated), sort the resulting knot locations internally, and build the design as X <- cbind(splineX(..., intercept=intercept), x_base). Invalid chromosomes/inputs return Inf.

Value

A single numeric value: the requested information criterion. Lower is better. Returns Inf for invalid chromosomes/inputs.

See Also

[varyknotsIC()], [splineX()], bs, ns

Examples

```
library(MASS)
y <- mcycle$accel
x <- mcycle$times
x_unique <- sort(unique(x))
# chromosome encoding 5 interior knot indices with sentinel:
chrom <- c(5, 24, 30, 46, 49, 69, length(x_unique) + 1)
fixknotsIC(chrom,
  y = y, x = x, x_unique = x_unique,
  fixedknots = 5, ic_method = "BIC"
)
```

---

gareg-class *S4 Class Definition for 'gareg'*

Description

S4 Class for Genetic Algorithm-Based Regression

S4 container for GA-based regression/changepoint tasks. Holds the GA backend fit and a normalized summary of the best solution.

Slots

`call` The matched call that created the object.

`N` The effective size of the x grid used for knot search (i.e., ‘length(x_unique)‘), typically the number of unique ‘x‘.

`call` language. The original call.

`method` character. One of "varyknots", "fixknots", "subset".

`N` numeric. Length of ‘x_unique‘ used by the GA (also ‘sentinel-1‘).

`objFunc` functionOrNULL. Objective function used.

`gaMethod` character. GA engine name ("cptga","cptgaisl","ga","gaisl").

`gaFit` Backend GA fit object (union of classes from GA and changepointGA).

`ctrl` listOrNULL. Control list used to run the GA (if stored by caller).

`fixedknots` numericOrNULL. Fixed number of interior knots (‘m‘) for fixed-knots mode, or NULL.

`minDist` numeric. Minimum distance between adjacent changepoints.

`polydegree` numericOrNULL. Spline degree for default objectives.

`type` character. One of ‘c("ppolys", "ns", "bs")‘ indicating piecewise polynomials, natural cubic, or B-spline.

`intercept` logical. Whether the spline basis included an intercept column.

`subsetSpec` listOrNULL. Constraints for subset selection (unused for knots).

`featureNames` character. Candidate feature names (subset tasks).

`bestFitness` numeric. Best fitness value found.

`bestChrom` numeric. Raw best chromosome returned by the backend (may include a sentinel equal to ‘N+1‘ and optional padding).

`bestnumbsol` numeric. Count of selected elements (e.g., ‘m‘ for knots).

`bestsol` numericOrChara. For knots: the ‘m‘ interior indices (pre-sentinel); for subset: mask/indices/names.

See Also

[gareg_knots], [cptgaControl]

`gareg-methods` *Show and summary methods for* `gareg`

Description

- `show(object)`: Compact header with call, engine, and N.
- `summary(object, ...)`: GA settings (when available) and best solution.

Usage

```
## S4 method for signature 'gareg'
show(object)

## S4 method for signature 'gareg'
summary(object, ...)
```

Arguments

object	A "gareg" object.
...	Currently unused.

Details

Methods for displaying and summarizing ‘gareg‘ objects

Value

show: invisible NULL. summary: invisibly returns object.

See Also

gareg-class

---

gareg_knots *Genetic-Algorithm–based Optimal Knot Selection*

Description

Runs a GA-based search for changepoints/knots and returns a compact "gareg" S4 result that stores the backend GA fit ("cptga" or "cptgaisl") plus the essential run settings.

Usage

```
gareg_knots(
  y,
  x,
  ObjFunc = NULL,
  fixedknots = NULL,
  minDist = 3L,
  degree = 3L,
  type = c("ppolys", "ns", "bs"),
  intercept = TRUE,
  gaMethod = "cptga",
  cptgactrl = NULL,
  monitoring = FALSE,
  seed = NULL,
  ...
)
```

Arguments

y	Numeric vector of responses (length N).
x	Optional index/time vector aligned with y. If missing, it defaults to seq_along(y). Used to derive x_unique (candidate knot positions) and passed to the objective function; the GA backend itself does not use x directly.
ObjFunc	Objective function or its name. If NULL, a default is chosen: • fixknotsIC when fixedknots is supplied; • varyknotsIC otherwise. A custom function must accept the chromosome and needed data via named arguments (see the defaults for a template function).
fixedknots	NULL (varying-knots search) or an integer giving the number of interior knots for a fixed-m search. If non-NULL, the method is "fixknots" and specialized operators are injected unless overridden in cptgactrl.
minDist	Integer minimum distance between adjacent changepoints. If omitted (missing() or NULL), the value in cptgactrl is used. If supplied here, it overrides the control value.
degree	Integer polynomial degree for "ppolys" and "bs". Ignored for "ns" (always cubic). Must be provided for "ppolys" and "bs".
type	One of c("ppolys", "ns", "bs"): piecewise polynomials, natural cubic, or B-spline. See splineX. The first option of 'ppolys' is taken by default.
intercept	Logical; include intercept column where applicable. Default: TRUE.
gaMethod	GA backend to call: function or name. Supports "cptga" (single population) and "cptgaisl" (islands).
cptgactrl	Control list built with cptgaControl() (or a named list of overrides). When gaMethod = "cptgaisl", island-specific knobs like numIslands and maxMig are recognized. Other genetic algorithm parameters can be found in cptga and cptgaisl.
monitoring	Logical; print short progress messages (also forwarded into the backend control).
seed	Optional RNG seed; also stored into the backend control.
...	Additional arguments passed to the GA backend. If the backend does not accept ..., unknown arguments are silently dropped (the call is filtered against the backend formals).

Details

Engine selection and controls. The function detects the engine from gaMethod and constructs a matching control via cptgaControl():

- "cptga" uses .cptga.default.
- "cptgaisl" uses .cptgaisl.default (supports numIslands, maxMig, etc.).
- see other details in cptga and cptgaisl.

Top-level monitoring, seed, and minDist given to gareg_knots() take precedence over the control list.

Fix-knots operators. When fixedknots is provided and the control does not already override them, the following operators are injected: Popinitial_fixknots, crossover_fixknots, mutation_fixknots.

Spline basis options. To build spline design matrices (via splineX):

- type = "ppolys": Degree-d regression spline via truncated-power piecewise polynomials.
- type = "ns": Degree-3 natural cubic spline with zero second-derivative at boundaries.
- type = "bs": Degree-d B-spline basis (unconstrained).

Value

An object of class "gareg" with key slots:

- call, method ("varyknots" or "fixknots"), N.
- objFunc, gaMethod, gaFit (class "cptga" or "cptgaisl"), ctrl.
- fixedknots, minDist, polydegree, type, intercept.
- bestFitness, bestChrom, bestnumbsol, bestsol.

Use summary(g) to print GA settings and the best solution (extracted from g@gaFit); show(g) prints a compact header.

Argument precedence

Values are combined as *control* < *core* < .... That is, cptgactrl provides defaults, then core arguments from gareg_knots() override those, and finally any matching names in ... override both.

See Also

cptgaControl, changepointGA::cptga, changepointGA::cptgaisl, fixknotsIC, varyknotsIC

Examples

```
set.seed(1)
N <- 120
y <- c(rnorm(40, 0), rnorm(40, 3), rnorm(40, 0))
x <- seq_len(N)

# 1) Varying-knots with single-pop GA
g1 <- gareg_knots(
  y, x,
  minDist = 5,
  gaMethod = "cptga",
  cptgactrl = cptgaControl(popSize = 150, pcrossover = 0.9, maxgen = 500)
)
summary(g1)

# 2) Fixed knots (operators auto-injected unless overridden)
g2 <- gareg_knots(
```

```
  y, x,
  fixedknots = 5,
  minDist = 5
)
summary(g2)

# 3) Island GA with island-specific controls
g3 <- gareg_knots(
  y, x,
  gaMethod = "cptgaisl",
  minDist = 6,
  cptgactrl = cptgaControl(
    engine = "cptgaisl",
    numIslands = 8, maxMig = 250,
    popSize = 120, pcrossover = 0.9
  )
)
summary(g3)
```

---

gareg_subset *Genetic-Algorithm Best Subset Selection (GA / GAISL)*

Description

Runs a GA-based search over variable subsets using a user-specified objective (default: subsetBIC) and returns a compact "gareg" S4 result with method = "subset". The engine can be ga (single population) or gaisl (islands), selected via gaMethod.

Usage

```
gareg_subset(
  y,
  X,
  ObjFunc = NULL,
  gaMethod = "ga",
  gacontrol = NULL,
  monitoring = FALSE,
  seed = NULL,
  ...
)
```

Arguments

y	Numeric response vector (length n).
X	Numeric matrix of candidate predictors (n rows by p columns).

ObjFunc	Objective function or its name. Defaults to subsetBIC. The objective must accept as its first argument a binary chromosome (0/1 mask of length p) and may accept additional arguments passed via .... By convention, subsetBIC returns *negative* BIC, so the GA maximizes fitness.
gaMethod	GA backend to call: "ga" or "gaisl" (functions from package **GA**), or a GA-compatible function with the same interface as ga.
gacontrol	Optional named list of GA engine controls (e.g., popSize, maxiter, run, pcrossover, pmutation, elitism, seed, parallel, keepBest, monitor, ...). These are passed to the GA engine, not to the objective.
monitoring	Logical; if TRUE, prints a short message and (if not supplied in gacontrol) sets monitor = GA::gaMonitor for live progress.
seed	Optional RNG seed (convenience alias for gacontrol$seed).
...	Additional arguments forwarded to ObjFunc (not to the GA engine). For subsetBIC these typically include family, weights, offset, and control.

Details

The fitness passed to **GA** is ObjFunc itself. Because the engine expects a function with signature f(chrom, ...), your ObjFunc must interpret chrom as a 0/1 mask over the columns of X. The function then computes a score (e.g., negative BIC) using y, X, and any extra arguments supplied via ....

With the default subsetBIC, the returned value is -BIC, so we set max = TRUE in the GA call to maximize fitness. If you switch to an objective that returns a quantity to *minimize*, either negate it in your objective or change the engine setting to max = FALSE.

Engine controls belong in gacontrol; objective-specific options belong in .... This separation prevents accidental name collisions between GA engine parameters and objective arguments.

Value

An object of S4 class "gareg" (with method = "subset") containing:

- call – the matched call.
- N – number of observations.
- objFunc – the objective function used.
- gaMethod – "ga" or "gaisl".
- gaFit – the GA fit object returned by **GA** (if your class allows it).
- featureNames – column names of X (or empty).
- bestFitness – best fitness value (GA::ga@fitnessValue).
- bestChrom – c(m, idx): number of selected variables and their indices.
- bestnumbsol – m, number of selected variables.
- bestsol – vector of selected column indices in X.

See Also

subsetBIC, ga, gaisl

Examples

```
if (requireNamespace("GA", quietly = TRUE)) {
  set.seed(1)
  n <- 100
  p <- 12
  X <- matrix(rnorm(n * p), n, p)
  y <- 1 + X[, 1] - 0.7 * X[, 4] + rnorm(n, sd = 0.5)

  # Default: subsetBIC (Gaussian − negative BIC), engine = GA::ga
  fit1 <- gareg_subset(y, X,
    gaMethod = "ga",
    gacontrol = list(popSize = 60, maxiter = 80, run = 40, parallel = FALSE)
  )
  summary(fit1)

  # Island model: GA::gaisl
  fit2 <- gareg_subset(y, X,
    gaMethod = "gaisl",
    gacontrol = list(popSize = 40, maxiter = 60, numIslands = 4, parallel = FALSE)
  )
  summary(fit2)

  # Logistic objective (subsetBIC handles GLM via ...):
  ybin <- rbinom(n, 1, plogis(0.3 + X[, 1] - 0.5 * X[, 2]))
  fit3 <- gareg_subset(ybin, X,
    gaMethod = "ga",
    family = stats::binomial(), # <- passed to subsetBIC via ...
    gacontrol = list(popSize = 60, maxiter = 80, parallel = FALSE)
  )
  summary(fit3)
}
```

---

mutation_fixknots *Mutation Operator (Fixed-Knots)*

Description

Replaces a child with a fresh feasible sample having the same m, drawn by selectTau_uniform_exact.

Usage

```
mutation_fixknots(child, p.range = NULL, minDist, Pb, lmax, mmax, N)
```

Arguments

child	Current chromosome (its first entry defines m).
p.range, Pb	Unused placeholders (kept for compatibility).

`minDist`	Integer minimum spacing.
`lmax`, `mmax`	Integers; chromosome length and maximum m (unused).
`N`	Integer series length.

Value

New feasible chromosome with the same m.

See Also

crossover_fixknots

`nhtr2005` *Northern Hemisphere temperature reconstruction*

Description

The 2,000-Year Northern Hemisphere Temperature Reconstruction dataset (Moberg et al., 2005) provides annual temperature anomalies for the Northern Hemisphere from AD 1 to 1979, relative to the 1961–1990 mean. The reconstruction combines high-resolution proxy data (e.g., tree rings) with low-resolution proxies (e.g., sediments) using a wavelet-based method to capture variability across multiple time scales.

Usage

```
nhtr2005
```

Format

A data frame with 2 variables:

Year Year AD

Temp Temperature anomaly relative to the 1961–1990 mean

Source

Moberg A, Sonechkin DM, Holmgren K, Datsenko NM, Karlén W. 2,000-Year Northern Hemisphere Temperature Reconstruction. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series #2005-019. NOAA/NGDC Paleoclimatology Program, Boulder, Colorado, USA.

References

Moberg A, Sonechkin DM, Holmgren K, Datsenko NM, Karlén W. (2005). Highly variable Northern Hemisphere temperatures reconstructed from low- and high-resolution proxy data. *Nature*, 433(7026), 613–617.

`Popinitial_fixknots` *Fixed-Knots Population Initializer*

Description

Initializes a population matrix for the fixed-knots GA. Each column is a feasible chromosome sampled by selectTau_uniform_exact.

Usage

```
Popinitial_fixknots(
  popSize,
  prange = NULL,
  N,
  minDist,
  Pb,
  mmax,
  lmax,
  fixedknots
)
```

Arguments

`popSize`	Integer; number of individuals (columns).
`prange`	Optional hyperparameter range (unused here).
`N`	Series length.
`minDist`	Integer minimum spacing between adjacent changepoints.
`Pb`	Unused placeholder (kept for compatibility).
`mmax`, `lmax`	Integers; maximum number of knots and chromosome length.
`fixedknots`	Integer; number of knots to place.

Value

Integer matrix of size `lmax` x `popSize`; each column is a chromosome `c(m, tau_1, ..., tau_m, N+1, ...)`.

See Also

selectTau_uniform_exact, gareg_knots

`selectTau_uniform_exact` *Exact Uniform Sampler of Feasible Changepoints*

Description

Samples m ordered changepoint indices uniformly from all feasible configurations on `1:N` subject to a minimum spacing `minDist`. Encodes the result as a chromosome for downstream GA operators.

Usage

```
selectTau_uniform_exact(N, m, minDist, lmax)
```

Arguments

`N`	Integer series length.
`m`	Integer number of changepoints to place.
`minDist`	Integer minimum spacing between adjacent changepoints.
`lmax`	Integer chromosome length.

Value

Integer vector length `lmax`: `c(m, tau_1, ..., tau_m, N+1, 0, 0, ...)`.

See Also

Popinitial_fixknots, mutation_fixknots

`splineX` *Build spline design matrices (piecewise polynomials, natural cubic, B-spline)*

Description

Unified wrapper to generate spline covariates for three common cases:

- `type = "ppolys"`: Degree-d regression spline via truncated-power **piecewise polynomials** (uses internal `tp_basis()`).
- `type = "ns"`: Degree-3 **natural cubic spline**; enforces $f''(a) = f''(b) = 0$ at the boundary.
- `type = "bs"`: Degree-d **B-spline** basis (unconstrained).

Usage

```
splineX(
  x,
  knots,
  degree = NULL,
  type = c("ppolys", "ns", "bs"),
  intercept = TRUE
)
```

Arguments

x	Numeric vector of predictor values.
knots	Numeric vector of interior knots.
degree	Integer polynomial degree for "ppolys" and "bs". Ignored for "ns" (always cubic). Must be provided for "ppolys" and "bs".
type	One of c("ppolys", "ns", "bs").
intercept	Logical; include intercept column where applicable. Default: 'TRUE'.

Details

Knots are sorted, no-duplicated, and any knots outside range(x) are dropped with a warning. For type = "ns", degree is ignored (natural splines are cubic).

Value

A numeric design matrix. Attributes are attached:

- "knots" — the interior knots used
- "boundary" — range(x)
- "degree" — effective degree (i.e., 3 for "ns")
- "type" — the requested spline type

See Also

bs, ns

Examples

```
set.seed(1)
x <- sort(rnorm(100))
k <- quantile(x, probs = c(.25, .5, .75))

# 1) Piecewise polynomials (degree 3)
X_pp <- splineX(x, knots = k, degree = 3, type = "ppolys", intercept = TRUE)
dim(X_pp) # n x ((3+1) + 3) = n x 7

# 2) Natural cubic spline (cubic, degree ignored)
X_ns <- splineX(x, knots = k, type = "ns", intercept = TRUE)
```

```
# 3) B-spline basis (degree 3)
X_bs <- splineX(x, knots = k, degree = 3, type = "bs", intercept = TRUE)

# Fit without a duplicated intercept:
# fit <- lm(y ~ 0 + X_pp)
```

---

subsetBIC *Unified BIC-style Objective for Subset Selection (GLM & Gaussian)*

Description

Computes a BIC-like criterion for a chromosome that encodes a variable subset. The same expression

$$\mathrm{BIC} = n \log(\mathrm{rss\_like}/n) + k \log n$$

is used for all families, where:

- For Gaussian with identity link, `rss_like` is the residual sum of squares (RSS), computed via a fast `.lm.fit`.
- For other GLM families, `rss_like` is the residual *deviance* from `glm.fit`.

The effective parameter count k includes the intercept.

Usage

```
subsetBIC(
  subset_bin,
  y,
  X,
  family = stats::gaussian(),
  weights = NULL,
  offset = NULL,
  control = stats::glm.control()
)
```

Arguments

`subset_bin`	Integer/numeric 0–1 vector (length `ncol(X)`); 1 means the corresponding column of `X` is included in the model.
`y`	Numeric response vector of length `n`.
`X`	Numeric matrix of candidate predictors; columns correspond to variables.
`family`	A GLM family object (default `stats::gaussian()`).
`weights`	Optional prior weights (passed to `glm.fit`).
`offset`	Optional offset (passed to `glm.fit`).
`control`	GLM fit controls; default `stats::glm.control()`.

Details

The chromosome `subset_bin` is a *binary* vector (0/1 by column), indicating which predictors from `X` are included. The design matrix always includes an intercept. Rank-deficient selections return `Inf` (which the GA maximizer treats as a very poor score). The value returned is **-BIC** so that GA engines can *maximize* it.

Value

A single numeric value: **-BIC**. Larger is better for GA maximizers. Returns `Inf` for rank-deficient designs.

See Also

`glm.fit`, `glm.control`, `.lm.fit`

---

varyknotsIC — *Information criterion for spline regression with a variable number of knots*

Description

Evaluates an information criterion (BIC, AIC, or AICc) for a regression of `y` on a spline basis of `x` where the number and locations of interior knots are encoded in the chromosome. Designed for use as a GA objective/fitness function. The spline basis is constructed via [splineX()].

Usage

```
varyknotsIC(
  knot_bin,
  plen = 0,
  y,
  x,
  x_unique,
  x_base = NULL,
  degree = 3L,
  type = c("ppolys", "ns", "bs"),
  intercept = TRUE,
  ic_method = "BIC"
)
```

Arguments

`knot_bin`	Integer vector (chromosome). Gene 1 stores m, the number of interior knots. Genes 2:(1+m) are indices into `x_unique` selecting the m interior knots, followed by a sentinel equal to `length(x_unique)+1`. Only genes strictly before the first occurrence of `length(x_unique)+1` are treated as interior indices; genes after the sentinel are ignored. Interior indices must be in `2:(length(x_unique)-1)`, finite, and non-duplicated.

plen	Unused placeholder kept for API compatibility; ignored.
y	Numeric response vector of length n.
x	Numeric predictor (same length as y) on which the spline basis is constructed.
x_unique	Optional numeric vector of unique candidate knot locations. If missing or NULL, defaults to sort(unique(x)). Must have at least three values (two boundaries + one interior) to allow any knots.
x_base	Optional matrix (or vector) of additional covariates to include linearly alongside the spline basis; coerced to a matrix if supplied.
degree	Integer polynomial degree for type="ppolys" and type="bs" (default 3L). Ignored for type="ns" (always cubic).
type	One of c("ppolys","ns","bs"); forwarded to [splineX()].
intercept	Logical; forwarded to [splineX()]. For $m > 0$, the spline block is splineX(..., intercept=intercept) and *no explicit 1-column* is added here; for $m = 0$, an explicit intercept is added via cbind(1, x_base). Set intercept=FALSE if you plan to add your own 1-column.
ic_method	Which information criterion to return: "BIC", "AIC", or "AICc".

Details

If $m = 0$, the model is a pure-linear baseline using only an intercept and x_base: X <- cbind(1, x_base) (no spline terms). For $m > 0$, the spline block is built with [splineX()] using the selected interior knots, with X <- cbind(splineX(..., intercept=intercept), x_base).

The criteria are computed as:

$$\mathrm{BIC} = n \log(\mathrm{SSRes}/n) + p \log n,$$

$$\mathrm{AIC} = n \log(\mathrm{SSRes}/n) + 2p,$$

$$\mathrm{AICc} = n \log(\mathrm{SSRes}/n) + 2p + \frac{2p(p+1)}{n-p-1},$$

where SSRes is the residual sum of squares and p is the number of columns in the design matrix X.

Value

A single numeric value: the requested information criterion (lower is better). Returns Inf for invalid chromosomes/inputs.

Note

This function allows $m = 0$ (no spline terms) so that the GA can compare against a pure-linear baseline (intercept + x_base). Spacing constraints (e.g., minimum distance between indices) should be enforced by the GA operators or an external penalty.

See Also

[fixknotsIC()], [splineX()], bs, ns

Examples

```
## Example with 'mcycle' data (MASS)
# y <- mcycle$accel; x <- mcycle$times
# x_unique <- sort(unique(x))
# chrom <- c(5, 24, 30, 46, 49, 69, length(x_unique) + 1)
# varyknotsIC(chrom, y=y, x=x, x_unique=x_unique,
#             type="ppolys", degree=3, ic_method="BIC")
```

Index