Package: gaQSAR (via r-universe)

June 24, 2026

Type Package

Title QSAR Modelling Using Genetic Algorithm Based Variable Selection

Version 1.2.3

Description Implements genetic algorithm-based variable selection for building quantitative structure-activity relationship (QSAR) models. The package provides a workflow for selecting optimal predictor subsets from large descriptor spaces using leave-one-out cross-validation (LOOCV) with Q2 as the fitness criterion. Features include automatic handling of multicollinearity via variance inflation factor (VIF) thresholding, customizable genetic algorithm operators, and diagnostic tools for model evaluation. Supports both training set optimization and external validation, plus nested (double) cross-validation for unbiased performance estimation and predictor stability diagnostics. Built-in visualization functions include Q2 curves and Williams plots to assess model applicability domain. The method is demonstrated in papers predicting antibacterial activity by Araya-Cloutier et al. (2018) <doi:10.1038/s41598-018-27545-4> and Kalli et al. (2021) <doi:10.1038/s41598-021-92964-9>.

License GPL-3

Encoding UTF-8

Imports GA, future, future.apply, ggplot2, ggrepel, stats, scales, prospectr, reshape2

Suggests knitr, rmarkdown, QSARdata

VignetteBuilder knitr

URL https://github.com/joshageman/gaQSAR

BugReports https://github.com/joshageman/gaQSAR/issues

NeedsCompilation no

Author Jos Hageman [aut, cre]

Maintainer Jos Hageman <jos.hageman@wur.nl>

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-06-24 08:20:07 UTC

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

RemoteRef HEAD

RemoteSha fca1d03eed57c6cdf2e6d8b17e9d9ed9b7be5021

Contents

createBestFitnessPlot *Plot best fitness per generation*

Description

Builds a simple line plot of the GA best fitness value per generation. Works with both gaQSAR and gaQSAR_dcv objects. For nested CV objects (gaQSAR_dcv), one series is shown per outer fold (each inner GA run).

Usage

```
createBestFitnessPlot(object, title = NULL)
```

Arguments

object	A gaQSAR object or a gaQSAR_dcv object.
title	Optional character plot title. If missing, a default title is chosen based on the object type.

Value

A ggplot2 object.

createDCVTrainingMetricsPlot *Plot training metrics (R2, R2adj, Q2) versus model size for nested CV runs*

Description

Summarizes training metrics across outer folds for one or multiple gaQSAR_dcv objects and plots mean +/- SE versus the number of predictors in a single panel. Metrics share axes and are distinguished by color/shape.

Usage

```
createDCVTrainingMetricsPlot(
  dcvResults,
  metrics = c("R2", "R2adj", "Q2"),
  title = "",
  includeOuterQ2 = FALSE
)
```

Arguments

dcvResults	A gaQSAR_dcv object or a list of gaQSAR_dcv objects. Use a list when you ran nested CV for multiple numberOfVariables.
metrics	Character vector of metrics to plot. Allowed values: "R2", "R2adj", "Q2" (inner). Use includeOuterQ2 = TRUE to add the outer-fold Q2 as a separate series.
title	Optional character plot title.
includeOuterQ2	Logical; if TRUE, append the outer Q2 value from each gaQSAR_dcv as an additional metric in the plot. Default FALSE.

Details

SE is computed across outer folds with status "ok".

Value

A ggplot2 object.

---

createDCVWilliamsPlot *Williams plot for double cross-validation diagnostics*

Description

Build a Williams plot for a gaQSAR_dcv run using diagnostics collected across training folds. Optionally aggregate per object (mean/median) or show raw points (no aggregation).

Usage

```
createDCVWilliamsPlot(
  dcvResult,
  residualThreshold = 2.5,
  aggregation = c("mean", "median", "none"),
  colorBy = c("objectId", "fold", "none", "object"),
  label = "",
  labelOutliers = c("rowNumber", "rowName", "none")
)
```

Arguments

dcvResult	A gaQSAR_dcv object produced by gaDoubleCrossValidation().
residualThreshold	Numeric threshold for standardized residual bands. Default is 2.5.
aggregation	Character scalar: "mean", "median", or "none" (raw points). Default is "mean".
colorBy	Character scalar controlling color mapping: "objectId", "fold", or "none". "object" is accepted as a backward-compatible alias for "objectId". Default is "objectId".

label	Optional character string appended to the plot title.
labelOutliers	Character scalar controlling outlier labeling: "rowNumber", "rowName", or "none". Outliers are defined as points with high leverage (> h*) and/or high residual (> residualThreshold). Default is "rowNumber".

Value

A ggplot2 object.

---

createQ2Plot *Plot Q2 versus number of predictors*

Description

Create a diagnostic plot of cross-validated performance against model size. The figure shows Q2 for the training set (LOOCV) and, when available, the external validation set as a function of the number of selected predictors. Use this for visual inspection of GA-based variable selection results and to compare model parsimony versus predictive ability.

Usage

```
createQ2Plot(output, label = "")
```

Arguments

output	A gaQSAR object or a list of gaQSAR objects. Each object must contain named numeric entries numVar, Q2Loocv, and optionally Q2Ext. If Q2Ext is absent, only the training curve is drawn.
label	Character string used as the plot title (set to "" for no title).

Details

Lines are drawn only for groups that contain two or more points; otherwise individual points are shown. Point shape and color encode whether values originate from the training (Q2Loocv) or external (Q2Ext) set. The x-axis shows the number of predictors; tick labels are limited to integers.

Value

Returns a Q2 versus number of predictors plot.

See Also

gaVariableSelection(), Q2(), createWilliamsPlot()

createWilliamsPlot *Create Williams plots for QSAR model diagnostics*

Description

Creates Williams plots for one gaQSAR object or for a list of gaQSAR objects. A Williams plot shows standardized residuals against leverage and is commonly used to inspect outliers and the applicability domain of a QSAR model.

Usage

```
createWilliamsPlot(
  output,
  xtrain,
  ytrain,
  xtest,
  ytest,
  label = "",
  residualThreshold = 2.5,
  labelPoints = c("outliers", "all", "none")
)
```

Arguments

output	A gaQSAR object or a list of gaQSAR objects. Each object must contain importantPredictors, either as predictor names or as column indices referring to xtrain.
xtrain	A data.frame or matrix with training predictors. Row names are used as object identifiers when available.
ytrain	Numeric vector with training responses, aligned with the rows of xtrain.
xtest	A data.frame or matrix with validation predictors. It must contain the selected predictor columns. Row names are used as object identifiers when available.
ytest	Numeric vector with validation responses, aligned with the rows of xtest.
label	Optional character string appended to the plot title.
residualThreshold	Numeric threshold for standardized residuals. The default is 2.5.
labelPoints	Character scalar controlling point labels: "outliers" labels points with high standardized residuals and/or high leverage, "all" labels all points, and "none" suppresses point labels. The default is "outliers".

Details

For each model, the selected predictors are taken from importantPredictors. The model is refitted on xtrain and ytrain. Training leverages are computed from the fitted design matrix. Validation leverages are computed from the validation design matrix using the inverse information matrix from the training data.

Training standardized residuals are computed as `residual / (sigma * sqrt(1 - h))`. Validation residuals are standardized as prediction residuals using `residual / (sigma * sqrt(1 + h))`, where `h` is the leverage of the validation object relative to the training design. This avoids undefined residuals when validation leverages are larger than one.

Horizontal reference lines are drawn at 0 and at `+/- residualThreshold`. The leverage threshold is `3 * p / n`, where `p` is the number of model coefficients including the intercept and `n` is the number of training observations.

Points are treated as Williams plot outliers when they have an absolute standardized residual larger than `residualThreshold` and/or a leverage larger than the leverage threshold.

Value

The input object, with each `gaQSAR` object augmented by: - `williamsData`: data.frame with object identifiers, leverages, residuals and dataset type. - `williamsPlot`: a `ggplot2` Williams plot. - `williamsOutliers`: counts of high-residual and high-leverage objects

See Also

`Q2()`, `predictOOBObjects()`

`gaDoubleCrossValidation`

Nested (double) cross-validation for GA-based variable selection

---

Description

Performs nested cross-validation combining an outer CV loop (for final model assessment) with an inner CV loop (used by the GA for fitness evaluation during variable selection). The inner loop always uses LOOCV via `singleCV` as the GA fitness. The outer loop may be leave-one-out or k-fold (default), providing an unbiased estimate of outer-loop predictive performance and stability analysis of selected predictors.

Usage

```
gaDoubleCrossValidation(
  x,
  y,
  outerMethod = "kfold",
  outerK = 5,
  seed = NULL,
  residualThreshold = 2.5,
  verbose = FALSE,
  ...
)
```

Arguments

x	A data.frame or matrix of predictors (rows = observations, columns = candidate variables).
y	Numeric response vector aligned with rows of x.
outerMethod	Character; outer CV method: "loo" for leave-one-out or "kfold" for k-fold cross-validation (default).
outerK	Integer; number of folds for outer k-fold CV (only used if outerMethod = "kfold"). Defaults to 5.
seed	Integer; random seed for reproducible fold splits (especially relevant for k-fold CV). If NULL, fold splits are non-deterministic.
residualThreshold	Numeric; threshold for standardized residuals in Williams plot diagnostics.
verbose	Logical; if TRUE, print brief progress information per outer fold. Defaults to FALSE (silent operation).
...	Additional arguments forwarded to gaVariableSelection(), such as numberOfVariables, popSize, maxIter, pMutation, pCrossover, and other GA settings.

Details

For each outer fold:

1. Create outerTrain (all data except outerTest) and outerTest (hold-out fold).
2. Run gaVariableSelection() on outerTrain using inner CV for fitness evaluation.
3. Fit the selected lm model on outerTrain using selected predictors.
4. Predict on outerTest and store predictions and residuals.
5. Compute and store diagnostic measures (Williams plot data, VIF, etc.).

Two types of seeds are employed:

- The seed parameter (outer fold seed) controls reproducible creation of outer CV folds.
- The seeds parameter (passed via ... to gaVariableSelection) enables multiple GA runs for robustness within each fold. The best seed is tracked in foldSummaries$bestSeed.

Across all outer folds, summaries are computed:

- selectionFrequency: How often each predictor is selected across successful outer folds.
- selectionFrequencyAllFolds: Optional frequency over all outer folds, including failed folds.
- signStability: For each selected predictor, the proportion of times its coefficient is positive (vs negative) across folds.
- modelSizeDistribution: Distribution of numbers of selected predictors.
- perObject outlier/high-leverage frequencies based on Williams plot thresholds.

Value

An object of class "gaQSAR_dcv" (list) containing:

- call: The matched call.
- outer: List with elements `method`, `k`, `seed` (the outer fold seed for reproducibility), and `folds` (list of outer fold indices).
- outerPredictions: Numeric vector of length n (full dataset) with out-of-fold predictions.
- outerResiduals: Numeric vector of length n with outer residuals.
- yObserved: Original observed response vector (aligned with rows of `x`).
- outerQ2: Numeric; Q2 computed from outer predictions (NA if < 2 valid predictions).
- foldModels: List of gaQSAR objects (one per outer fold; NULL for failed folds).
- foldSummaries: Data.frame with one row per outer fold, including columns: nPredictors, selectedPredictors, innerQ2 (LOOCV fitness), trainingR2, trainingR2Adj, maxVif, bestSeed (the GA seed that produced the best model for that fold), status ("ok" or "failed"), errorMessage.
- foldDiagnostics: List with Williams plot diagnostic data per fold (NULL for failed folds); each element includes williamsData, hStar, residualThreshold.
- adThresholds: List describing applicability domain thresholds used: leverageCutoffMethod, leverageCutoff (NA if varies), residualCutoff, notes.
- selectionFrequency: Named numeric vector of predictor frequencies (0 to 1) computed over successful folds only.
- selectionFrequencyAllFolds: Named numeric vector of predictor frequencies (0 to 1) computed over all folds, including failed folds.
- signStability: Named numeric vector of positive sign frequency for selected predictors.
- modelSizeDistribution: Named integer vector of model size frequencies.
- perObjectOutlierFrequency: Numeric vector of length n with outlier frequencies.
- perObjectHighLeverageFrequency: Numeric vector of length n with high-leverage frequencies.

See Also

`gaVariableSelection()`, `singleCV()`, `Q2()`, `createWilliamsPlot()`

Examples

```
set.seed(42)
n <- 50
p <- 20
x <- matrix(rnorm(n * p), nrow = n)
colnames(x) <- paste0("X", seq_len(p))
y <- 2 * x[, 2] - 1.5 * x[, 5] + rnorm(n, sd = 0.5)

# Nested CV with LOO outer and inner LOOCV fitness
# NOTE: Settings below are example-only (fast runtime, not optimal).
# For real QSAR work, use larger population sizes, more generations, and multiple seeds.
```

```
# Good starting point are the default GA settings, or run an experimental design to
# tune GA settings for your dataset.
dcvFit <- gaDoubleCrossValidation(
  x = x,
  y = y,
  outerMethod = "kfold",
  outerK = 2,
  numberOfVariables = 3,
  popSize = 20,
  maxIter = 20,
  seeds = 1,
  interval = 5,
  verbose = TRUE
)

print(dcvFit)
summary(dcvFit)
plot(dcvFit)
plot(dcvFit, type = "selectionFrequency")
plot(dcvFit, type = "williamsFrequency")
```

`gaintegerMutation` *Integer-valued GA mutation operator*

Description

Mutate an integer-encoded chromosome by independently replacing each gene, with probability `object@pmutation`, by a new value drawn uniformly from `1:object@upper[i]`. Intended for use as the `mutation` function in `GA::ga()`.

Usage

```
gaintegerMutation(object, parent, ...)
```

Arguments

`object`	A `GA` object from the `GA` package. The operator reads `object@upper` (gene-wise upper bounds), `object@pmutation` (per-gene mutation probability), and `object@population` (matrix of chromosomes).
`parent`	Integer; row index of the parent chromosome in `object@population`.
`...`	Ignored; for compatibility with the `GA` package interface.

Details

Mutation decisions are drawn independently per gene using `stats::runif()`. When mutation occurs for gene `i`, a new integer in the range `[1, upper[i]]` is sampled with replacement via `sample.int()`.

Value

A numeric vector containing the mutated chromosome.

See Also

GA::ga

`gaintegerOnePointCrossover`
Integer-valued one-point crossover operator

Description

Perform one-point crossover on two parent chromosomes. A single crossover point is sampled uniformly over gene positions; genes to the left of the point (inclusive) are swapped between parents. Intended for use as the `crossover` function in `GA::ga()`.

Usage

```
gaintegerOnePointCrossover(object, parents, ...)
```

Arguments

`object`	A `GA` object from the `GA` package; uses `object@population` as the matrix of chromosomes.
`parents`	Integer vector of length 2 giving the row indices of the two parents in `object@population`.
`...`	Ignored; included for compatibility with the `GA` package interface.

Details

The function returns two children with identical lengths and integer encoding as the parents. The fitness values are not evaluated here and are returned as `NA_real_` placeholders, to be computed by the GA engine.

Value

A list with elements `children` (2 x nGenes integer matrix) and `fitness` (numeric vector of length 2 filled with `NA`).

See Also

GA::ga

gaintegerPopulation *Integer-valued GA population initializer*

Description

Create an initial population for integer-encoded chromosomes. For each gene position `j`, values are drawn independently and uniformly from the range `1:object@upper[j]`. Intended for use as the `population` function in `GA::ga()`.

Usage

```
gaintegerPopulation(object, ...)
```

Arguments

`object`	A `GA` object from the `GA` package; reads `object@upper` (gene-wise upper bounds) and `object@popSize` (number of individuals).
`...`	Ignored; present for compatibility with the `GA` package interface.

Details

The returned matrix has one row per individual and one column per gene. Sampling uses `sample.int()` with replacement.

Value

A numeric matrix with dimensions `popSize x length(upper)` containing the initial population.

See Also

GA::ga

gaintegerTwoPointCrossover *Integer-valued two-point crossover operator*

Description

Perform a two-point crossover on two parent chromosomes. Two crossover points are sampled uniformly, ordered as `p1 < p2`, and the outer segments (positions `1..p1` and `p2..end`) are swapped between parents. Intended for use as the `crossover` function in `GA::ga()`.

Usage

```
gaintegerTwoPointCrossover(object, parents, ...)
```

Arguments

object	A GA object from the GA package; uses object@population as the matrix of chromosomes.
parents	Integer vector of length 2 giving the row indices of the two parents in object@population.
...	Ignored; included for compatibility with the GA package interface.

Details

Returns two children with the same integer encoding and length as the parents. Fitness values are not computed here and are returned as NA_real_ placeholders.

Value

A list with elements children (2 x nGenes integer matrix) and fitness (numeric vector of length 2 filled with NA).

See Also

GA::ga

---

gaPermutationTest *Y-scrambling permutation test for GA-based variable selection*

Description

Performs a Y-scrambling permutation test for objects produced by gaQSAR or gaQSAR_dcv. In each permutation, the response vector is randomly permuted while the descriptor matrix is kept unchanged. The GA variable selection procedure is then repeated using the same settings as in the original analysis.

Usage

```
gaPermutationTest(
  object,
  x,
  nPermutations = 500,
  seed = NULL,
  validateSettings = FALSE,
  verbose = FALSE,
  workers = 1,
  ...
)
```

Arguments

object	A gaQSAR or gaQSAR_dcv object representing the true (unpermuted) result.
x	A data.frame or matrix of predictors (same as used in the original analysis).
nPermutations	Integer; number of permutations to perform. Default is 500.
seed	Integer; random seed for reproducibility of permutations. If NULL, permutations are non-deterministic.
validateSettings	Logical; if TRUE, re-runs the true (unpermuted) analysis using the stored settings to verify they reproduce the original result. If the re-run metric differs from the stored metric by more than 0.001, a warning is issued. Default is FALSE.
verbose	Logical; if TRUE, print progress information for each permutation. Default is FALSE.
workers	Integer; number of parallel workers to use for permutations. Use 1 to run sequentially. When workers > 1, the function uses a temporary future::multisession plan and future.apply::future_lapply() with future.seed = TRUE, restoring the previous plan on exit.
...	Currently unused.

Details

The permutation test works by:

1. Extracting the true performance metric (Q2 for gaQSAR, outer Q2 for gaQSAR_dcv).
2. For each permutation: scrambling y, re-running the analysis, and recording the permuted performance metric.
3. The empirical p-value is computed with a plus-one correction: (1 + nGreaterEqual) / (1 + nValid), where nGreaterEqual is the number of valid permutations with a performance metric greater than or equal to the observed metric, and nValid is the number of valid permutation runs.

Value

A list of class "gaQSAR_permTest" containing:

- trueMetric: Numeric; the true performance metric (Q2 or outer Q2).
- permutedMetrics: Numeric vector of length nPermutations with performance metrics from permuted runs.
- pValue: Numeric; empirical p-value with plus-one correction.
- nPermutations: Integer; number of permutations performed.
- objectType: Character; either "gaQSAR" or "gaQSAR_dcv".
- seed: Integer or NULL; random seed used.

See Also

gaVariableSelection(), gaDoubleCrossValidation()

Examples

```
# Simple example with gaQSAR object
set.seed(42)
n <- 50
p <- 20
x <- matrix(rnorm(n * p), nrow = n)
colnames(x) <- paste0("X", seq_len(p))
y <- 2 * x[, 2] - 1.5 * x[, 5] + rnorm(n, sd = 0.5)

# NOTE: Settings below are example-only (fast runtime, not optimal).
# For real QSAR work, use larger population sizes, more generations, and multiple seeds.
# Good starting point are the default GA settings, or run an experimental design to
# tune GA settings for your dataset.

fit <- gaVariableSelection(
  x = x,
  y = y,
  numberOfVariables = 3,
  popSize = 10,
  maxIter = 10,
  seeds = 1,
  interval = 5,
  verbose = TRUE
)

# For a meaningful permutation test, use a larger number of permutations (e.g., 100 or more).
permTest <- gaPermutationTest(
  object = fit,
  x = x,
  nPermutations = 2,  # small number for example; use 100+ for real tests
  verbose = TRUE
)
print(permTest)
```

`gaVariableSelection` *Genetic algorithm based variable selection for QSAR*

Description

Runs a genetic algorithm (GA) to select a fixed number of predictors using leave-one-out cross-validation (LOOCV) for stable optimization. The GA is executed for each seed in `seeds` for robustness, and the best run (highest LOOCV Q2) is returned.

Usage

```
gaVariableSelection(
  x,
  y,
```

```
  numberOfVariables = 4,
  pMutation = 0.2,
  pCrossover = 0.7,
  crossoverFunc = "gaintegerOnePointCrossover",
  popSize = 100,
  maxIter = 1300,
  elitism = 3,
  seeds = 1:5,
  interval = 50,
  verbose = FALSE
)
```

Arguments

x	A data.frame or matrix of predictors (rows = observations, columns = candidate variables).
y	Numeric response vector aligned with rows of x.
numberOfVariables	Integer; number of predictors to select.
pMutation	Numeric; mutation probability.
pCrossover	Numeric; crossover probability.
crossoverFunc	Character or function; crossover function used by the GA.
popSize	Integer; GA population size.
maxIter	Integer; maximum number of GA iterations.
elitism	Integer; number of elite individuals carried over.
seeds	Integer vector; RNG seeds to repeat GA runs for robustness.
interval	Integer; iteration interval for progress monitoring when verbose = TRUE.
verbose	Logical; if TRUE, prints progress information and GA settings.

Details

Fitness function: Uses LOOCV-based Q2 (via singleCV) as the GA fitness metric. LOOCV provides a deterministic, non-random evaluation landscape that enables stable variable selection, preventing fold-splitting variability from interfering with optimization. The nested CV structure (outer: validation, inner: LOOCV fitness) separates variable selection from performance assessment.

Robustness: For each seed, the GA runs with identical parameters. The run with the highest LOOCV Q2 is selected and returned. The best seed is tracked for reproducibility in permutation tests.

Value

An object of class "gaQSAR" containing the best GA run for the specified model size. The structure typically includes: numVar, importantPredictors, yTrain, model, R2Train, R2AdjTrain, Q2Loocv (LOOCV-based Q2 used by GA for fitness), VIF (variance inflation factors for each selected predictor), bestFitnessPerGeneration (best GA fitness value at each generation), bestSeed (the seed that produced the optimal solution), and gaSettings (list of GA parameters used, for reproducibility in permutation tests). Use plot.gaQSAR() to visualize any attached plots.

See Also

singleCV(), createQ2Plot(), createWilliamsPlot()

Examples

```
# This is a toy example for the documentation: GA settings are set so it runs fast.
# For real QSAR work: start with the  default GA settings, or run an experimental design
# to tune GA settings for your dataset.

set.seed(1)

# Create toy descriptor matrix (n compounds x p predictors)
n <- 40
p <- 20
x <- matrix(rnorm(n * p), nrow = n)

# Add names to mimic typical QSAR data structures (molecules + descriptor names)
rownames(x) <- paste0("mol_", seq_len(n))
colnames(x) <- paste0("X", seq_len(p))

# Create a synthetic response with a known signal in predictors 2 and 7
y <- 1.5 * x[, 2] - 0.8 * x[, 7] + rnorm(n, sd = 0.5)

# Run GA variable selection
# NOTE: Settings below are example-only (fast runtime, not optimal).
# For real QSAR work, use larger population sizes, more generations, and multiple seeds.
# Good starting point are the default GA settings, or run an experimental design to
# tune GA settings for your dataset.
fit <- gaVariableSelection(
  x = x,
  y = y,
  numberOfVariables = 2,       # target model size (number of selected predictors)
  popSize = 20,                # small population for speed (increase in real runs)
  maxIter = 20,                # few generations for speed (increase in real runs)
  seeds = 1,                   # single seed for a minimal example (use multiple seeds)
  interval = 5                # print progress every 5 generations when verbose=TRUE
)

# Methods for gaQSAR objects
print(fit)
summary(fit)
plot(fit, type = "predObs")  # observed vs predicted plot
```

`plot.gaQSAR` *Plot method for gaQSAR objects*

Description

Plot diagnostics for a gaQSAR result.

Usage

```
## S3 method for class 'gaQSAR'
plot(x, type = "all", ...)
```

Arguments

x	An object of class "gaQSAR" (returned from gaVariableSelection()).
type	Character scalar specifying which plot to create or print. Supported values are "all" (default), "fitness", "predObs" and "williams".
...	Additional arguments (currently unused).

Details

type = "all" prints available plots separately, one after another. The fitness plot is always created with createBestFitnessPlot(). The observed-versus-predicted plot is created from x$yTrain and, when available, x$yExt. The Williams plot use stored plot objects created by createWilliamsPlot().

Value

Invisibly returns the created or printed ggplot2 object for a single plot type, or a named list of plot objects for type = "all".

See Also

gaVariableSelection(), createWilliamsPlot()

---

plot.gaQSAR_dcv *Plot method for gaQSAR_dcv objects*

Description

Generate diagnostic plots for nested cross-validation results using ggplot2.

Usage

```
## S3 method for class 'gaQSAR_dcv'
plot(x, type = "outerPredObs", ...)
```

Arguments

x	An object of class "gaQSAR_dcv" (returned from gaDoubleCrossValidation()).
type	Character; plot type. Options: • "outerPredObs" (default): Observed vs predicted values with identity line. • "selectionFrequency": Bar plot of predictor selection frequencies. • "williamsFrequency": Per-object outlier and high-leverage frequencies. • "all": Generate all three plot types sequentially.
...	Additional arguments (currently unused).

Value

A ggplot2 object (or invisible NULL for type = "all").

See Also

print.gaQSAR_dcv(), summary.gaQSAR_dcv(), gaDoubleCrossValidation()

---

plot.gaQSAR_permTest *Plot method for gaQSAR_permTest objects*

Description

Creates a histogram of permuted metrics with a vertical line indicating the true metric value.

Usage

```
## S3 method for class 'gaQSAR_permTest'
plot(x, bins = 30, title = NULL, ...)
```

Arguments

x	A gaQSAR_permTest object.
bins	Integer; number of histogram bins. Default is 30.
title	Optional character plot title.
...	Additional arguments (unused).

Value

A ggplot2 object.

---

predictOOBObjects *Predict out-of-bag objects and compute external Q2*

Description

Compute external predictions for test objects using stored coefficient vectors (per selected model) and derive the external Q2 metric. For each element in output, predictor columns in x are matched by name to the coefficient names (excluding the intercept), an intercept term is prepended, and predictions are obtained via a matrix product. The resulting external Q2 and per-object predictions/residuals are attached to each element.

Usage

```
predictOOBObjects(output, x, y, verbose = FALSE)
```

Arguments

output	A gaQSAR object or a list of gaQSAR objects. Each object must contain a numeric coefficient vector model for the selected predictors, with the intercept in the first position and remaining names matching columns in x.
x	A data.frame or matrix with predictors in columns and observations in rows. Column names must cover the predictor names used in model (excluding the intercept). Row order must align with y.
y	Numeric vector of observed response values aligned with the rows of x.
verbose	Logical; if TRUE, prints the computed external Q2 for each model element.

Details

Predictor order does not matter: columns are matched by name to the coefficient vector (excluding the intercept). An intercept column of ones is automatically added before prediction.

Value

Returns an updated gaQSAR object (if a single object was supplied) or an updated list of gaQSAR objects, with each element augmented by:

- Q2Ext: numeric external Q2 computed by Q2(y, yHat).
- yExt: data.frame with columns y, yHat, and residual for test data.

See Also

Q2()

---

print.gaQSAR *Print method for gaQSAR objects*

Description

Print comprehensive information about a gaQSAR object returned from gaVariableSelection(). Displays model configuration, performance metrics, selected predictors, model coefficients, and data summaries.

Usage

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

Arguments

x	An object of class "gaQSAR" or a list of such objects (returned from gaVariableSelection()).
...	Additional arguments (currently unused).

Details

Prints detailed information including:

- Number of selected predictors and their indices/names
- Training R2, adjusted R2, and LOOCV Q2 metrics
- External validation Q2 (if available)
- OLS model coefficients (intercept and slopes)
- Variance Inflation Factors (VIF) for each selected predictor
- Summary statistics for training set residuals
- Summary statistics for external validation residuals (if available)
- Information about attached plots

Value

Invisibly returns the object.

See Also

summary.gaQSAR(), gaVariableSelection(), plot.gaQSAR()

---

print.gaQSAR_dcv *Print method for gaQSAR_dcv objects*

Description

Display a brief overview of nested cross-validation results.

Usage

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

Arguments

x	An object of class "gaQSAR_dcv" (returned from gaDoubleCrossValidation()).
...	Additional arguments (currently unused).

Value

Invisibly returns the object.

See Also

summary.gaQSAR_dcv(), gaDoubleCrossValidation()

print.gaQSAR_permTest *Print method for gaQSAR_permTest objects*

Description

Print method for gaQSAR_permTest objects

Usage

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

Arguments

x	A gaQSAR_permTest object.
...	Additional arguments (unused).

Value

Invisibly returns the input object.

Q2 *Compute Q2 (cross-validated R-squared)*

Description

Compute the Q2 statistic from observed and predicted values:

$$Q^2 = 1 - \frac{\sum (y_{obs} - y_{pred})^2}{\sum (y_{obs} - \bar{y}_{obs})^2}.$$

This function does not perform cross-validation; it only evaluates the formula for supplied predictions (e.g., from LOOCV or an external test set).

Usage

```
Q2(y, yhat)
```

Arguments

y	Numeric vector of observed response values.
yhat	Numeric vector of predicted response values, aligned with y.

Details

The vectors y and yhat must have the same length. The denominator is the total sum of squares of y; if y is constant this denominator is 0, making Q2 undefined (the result will be NaN or Inf). Missing values are not handled specially; if present, the result may be NA.

Value

A numeric scalar Q2 value. Values can be negative when predictive performance is poor and approach 1 for perfect predictions.

See Also

singleCV(), createQ2Plot(), predictOOBObjects()

Examples

```
y_obs <- c(2.1, 4.0, 5.9, 8.1, 10.2)
y_pred <- c(2.2, 3.8, 6.1, 7.9, 10.3)
Q2(y_obs, y_pred)
```

QSARMonitorFactory *GA monitor function for QSAR variable selection*

Description

Create a monitor callback for GA::ga() that prints concise progress lines at fixed intervals: generation number, best fitness value, number of selected variables, and their indices.

Usage

```
QSARMonitorFactory(interval = 50)
```

Arguments

interval	Integer; print progress every interval generations.

Details

The returned function closes over interval and is intended for the monitor argument of GA::ga(). It inspects the current GA object to obtain the best individual and reports its fitness and selected variable indices. Output is written to standard output via cat() and nothing is returned.

Value

A function suitable for the monitor argument of GA::ga().

See Also

GA::ga

`singleCV` *LOOCV Q2 fitness function for small datasets*

Description

Compute a leave-one-out cross-validated Q2 for a candidate set of predictors. Designed for use as a fitness function inside a genetic algorithm. Duplicate predictors are penalized with a large negative value, and candidate sets with variance inflation factor (VIF) above a threshold are rejected (fitness 0).

Usage

```
singleCV(predictors, x, y, vifThreshold = 5)
```

Arguments

`predictors`	Integer vector of 1-based predictor indices referring to columns of `x`.
`x`	Matrix or data.frame of predictors (rows = observations, columns = candidate variables).
`y`	Numeric response vector aligned with rows of `x`.
`vifThreshold`	Numeric; maximum allowed VIF (default 5).

Details

An intercept is added internally. VIF values are computed by regressing each selected predictor on the remaining selected predictors; if any VIF exceeds `vifThreshold`, the fitness is 0. LOOCV is performed via explicit refits for each left-out observation.

Value

A numeric scalar: Q2 if valid; 0 if the VIF constraint fails; or -100 if duplicate predictors are present.

splitUp *Split data into training and test sets*

Description

Split observations into training and test sets using either a Kennard-Stone split ("KS") based on sample representativeness or a simple random split. The function returns row indices for both sets.

Usage

```
splitUp(
  data,
  method = c("KS", "random"),
  trainPercentage = 0.8,
  pc = 0.95,
  verbose = FALSE
)
```

Arguments

data	Data frame or matrix with observations in rows (predictors and/or response may be present; only row indices are returned).
method	Character; one of "KS" (Kennard-Stone) or "random".
trainPercentage	Numeric in (0, 1); fraction of observations assigned to the training set.
pc	Numeric; proportion of variance to retain in the PCA step used by prospectr::kenStone() when method = "KS".
verbose	Logical; if TRUE, prints the chosen training and test IDs (row names).

Details

For method = "KS", the function requires the prospectr package and delegates to prospectr::kenStone() with k = round(trainPercentage * n) on a scaled version of data. For method = "random", rows are permuted and the first k indices are assigned to training.

Value

A list with integer vectors model (training indices) and test (test indices).

Examples

```
set.seed(123)
toy <- data.frame(x = rnorm(40), y = rnorm(40))
split <- splitUp(toy, method = "random", trainPercentage = 0.75)
str(split)
```

summary.gaQSAR *Summary method for gaQSAR objects*

Description

Produce a concise summary of a gaQSAR object returned from gaVariableSelection(). Displays key model configuration and performance metrics.

Usage

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

Arguments

object	An object of class "gaQSAR" (returned from gaVariableSelection()).
...	Additional arguments (currently unused).

Details

Prints a summary including:

- Number of selected predictors
- Selected predictor indices and names (if available)
- Training R2, adjusted R2, and LOOCV Q2
- External validation Q2 (if available)
- Model coefficients
- Variance Inflation Factors (VIF) for each selected predictor
- Residual statistics for training and validation sets

Value

An object of class "summary.gaQSAR" (invisibly).

See Also

print.gaQSAR(), gaVariableSelection(), plot.gaQSAR()

summary.gaQSAR_dcv *Summary method for gaQSAR_dcv objects*

Description

Produce a detailed summary of nested cross-validation results, including per-fold summaries and stability metrics.

Usage

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

Arguments

object	An object of class "gaQSAR_dcv" (returned from gaDoubleCrossValidation()).
...	Additional arguments (currently unused).

Value

An object of class "summary.gaQSAR_dcv" (invisibly).

See Also

print.gaQSAR_dcv(), gaDoubleCrossValidation()

summary.gaQSAR_permTest *Summary method for gaQSAR_permTest objects*

Description

Summary method for gaQSAR_permTest objects

Usage

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

Arguments

object	A gaQSAR_permTest object.
...	Additional arguments (unused).

Value

Invisibly returns the input object.

Index