Package 'corrselect'

May 8, 2026

Title Correlation-Based and Model-Based Predictor Pruning

Version 3.2.1

Description Provides functions for predictor pruning using association-based and model-based approaches. Includes corrPrune() for fast correlation-based pruning, modelPrune() for VIF-based regression pruning, and exact graph-theoretic algorithms (Eppstein–Löffler–Strash, Bron–Kerbosch) for exhaustive subset enumeration. Supports linear models, GLMs, and mixed models ('lme4', 'glmmTMB').

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

LinkingTo Rcpp

Imports Rcpp, S7, stats

Suggests GO.db, WGCNA, preprocessCore, impute, energy, minerva, lme4, glmmTMB, MASS, caret, car, carData, microbenchmark, igraph, Boruta, glmnet, corrplot, knitr, rmarkdown, testthat (>= 3.0.0), tibble, svglite, data.table

VignetteBuilder knitr

URL https://gillescolling.com/corrselect/

BugReports https://github.com/gcol33/corrselect/issues

Depends R (>= 3.5)

LazyData true

NeedsCompilation yes

Author Gilles Colling [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3070-6066>)

Maintainer Gilles Colling <gilles.colling051@gmail.com>

Repository CRAN

Date/Publication 2026-03-23 10:00:15 UTC

Contents

`corrselect-package` *corrselect: Correlation-Based and Model-Based Predictor Pruning*

Description

Provides tools for reducing multicollinearity in predictor sets through association-based and model-based approaches. The package offers both fast greedy algorithms for quick pruning and exact graph-theoretic algorithms for exhaustive subset enumeration.

Association-Based Pruning

These functions identify variable subsets where all pairwise correlations or associations remain below a user-defined threshold:

`corrPrune` Fast greedy pruning for numeric data

`corrSelect` Exhaustive enumeration for numeric data frames

`assocSelect` Exhaustive enumeration for mixed-type data (numeric, factor, ordered)

`MatSelect` Direct interface using a pre-computed correlation matrix

Model-Based Pruning

These functions use variance inflation factors (VIF) to iteratively remove collinear predictors from regression models:

`modelPrune` VIF-based pruning for lm, glm, lme4, and glmmTMB models

Algorithms

The exact enumeration functions (`corrSelect`, `assocSelect`, `MatSelect`) use two graph-theoretic algorithms:

Eppstein-Loffler-Strash (ELS) Recommended when using `force_in` constraints

Bron-Kerbosch Default algorithm, with optional pivoting for performance

Helpers

`corrSubset` Extract specific subsets from results

`CorrCombo` Class holding enumeration results

Author(s)

Maintainer: Gilles Colling <gilles.colling051@gmail.com> (ORCID) [copyright holder]

See Also

Vignettes: `vignette("quickstart", package = "corrselect")`, `vignette("advanced", package = "corrselect")`

---

as.data.frame.CorrCombo

Coerce CorrCombo to a Data Frame

Description

Converts a `CorrCombo` object into a data frame of variable combinations.

Usage

```
## S3 method for class 'CorrCombo'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A `CorrCombo` object.
`row.names`	Optional row names for the output data frame.
`optional`	Logical. Passed to `data.frame()`.
`...`	Additional arguments passed to `data.frame()`.

Value

A data frame where each row corresponds to a subset of variables. Columns are named `VarName01`, `VarName02`, ..., up to the size of the largest subset. Subsets shorter than the maximum length are padded with `NA`.

See Also

CorrCombo

Examples

```
set.seed(1)
mat <- matrix(rnorm(100), ncol = 10)
colnames(mat) <- paste0("V", 1:10)
res <- corrSelect(cor(mat), threshold = 0.5)
as.data.frame(res)
```

`assocSelect` *Select Variable Subsets with Low Association (Mixed-Type Data Frame Interface)*

Description

Identifies combinations of variables of any common data type (numeric, ordered factors, or unordered) factors—whose *pair-wise association* does not exceed a user-supplied threshold. The routine wraps `MatSelect()` and handles all pre-processing (type conversion, missing-row removal, constant-column checks) for typical data-frame/tibble/data-table inputs.

Usage

```
assocSelect(
  df,
  threshold = 0.7,
  method = NULL,
  force_in = NULL,
 method_num_num = c("pearson", "spearman", "kendall", "bicor", "distance", "maximal"),
  method_num_ord = c("spearman", "kendall"),
  method_ord_ord = c("spearman", "kendall"),
  ...
)
```

Arguments

`df`	A data frame (or tibble / data.table). May contain any mix of: • numeric / integer (treated as numeric) • ordered factors • unordered factors (character vectors are coerced to factors)
`threshold`	Numeric in $(0, 1)$. Maximum allowed pair-wise *absolute* association. Default `0.7`.
`method`	Character; the subset-search algorithm. One of `"els"` or `"bron-kerbosch"`. If `NULL` (default) the function selects automatically: ELS when `force_in` is supplied, otherwise Bron–Kerbosch.

`force_in`	Optional character vector or column indices specifying variables that must appear in every returned subset.
`method_num_num`	Association measure for numeric–numeric pairs. One of `"pearson"` (default), `"spearman"`, `"kendall"`, `"bicor"`, `"distance"`, or `"maximal"`.
`method_num_ord`	Association measure for numeric–ordered pairs. One of `"spearman"` (default) or `"kendall"`.
`method_ord_ord`	Association measure for ordered–ordered pairs. One of `"spearman"` (default) or `"kendall"`.
`...`	Additional arguments passed unchanged to `MatSelect()` (e.g., `use_pivot = TRUE` for Bron–Kerbosch).

Details

A single call can therefore screen a data set that mixes continuous and categorical features and return every subset whose internal associations are "sufficiently low" under the metric(s) you choose.

Rows containing `NA` are dropped with a warning; constant columns are treated as having zero association with every other variable.

The default association measure for each variable-type combination is:

numeric – numeric `method_num_num` (default `"pearson"`)

numeric – ordered `method_num_ord`

numeric – unordered `"eta"` (ANOVA η^2)

ordered – ordered `method_ord_ord`

ordered – unordered `"cramersv"`

unordered – unordered `"cramersv"`

All association measures are rescaled to $[0, 1]$ before thresholding. External packages are required for `"bicor"` (**WGCNA**), `"distance"` (**energy**), and `"maximal"` (**minerva**); an informative error is thrown if they are missing.

Value

A `CorrCombo` object containing:

- all valid subsets,
- their summary association statistics,
- metadata (algorithm used, rows kept, forced-in variables, etc.).

The object's `show()` method prints the association metrics that were *actually used* for this data set.

See Also

`corrSelect()`, `MatSelect()`, `corrSubset()`

Examples

```
set.seed(42)
df <- data.frame(
  height = rnorm(15, 170, 10),
  weight = rnorm(15, 70, 12),
  group  = factor(rep(LETTERS[1:3], each = 5)),
  score  = ordered(sample(c("low","med","high"), 15, TRUE))
)

## keep every subset whose internal associations <= 0.6
assocSelect(df, threshold = 0.6)

## use Kendall for all rank-based comparisons and force 'height' to appear
assocSelect(df,
            threshold       = 0.5,
            method_num_num  = "kendall",
            method_num_ord  = "kendall",
            method_ord_ord  = "kendall",
            force_in        = "height")
```

`bioclim_example` *Example Bioclimatic Data for Ecological Modeling*

Description

A simulated dataset with the 19 WorldClim bioclimatic variables (https://www.worldclim.org/data/bioclim.html) measured at 100 geographic locations, with species richness as the response variable. Variables are organized into correlated blocks representing temperature (BIO1-BIO11) and precipitation (BIO12-BIO19).

Usage

```
bioclim_example
```

Format

A data frame with 100 rows and 20 variables:

species_richness Integer. Number of species observed (response variable)

BIO1 Numeric. Annual Mean Temperature

BIO2 Numeric. Mean Diurnal Range

BIO3 Numeric. Isothermality

BIO4 Numeric. Temperature Seasonality

BIO5 Numeric. Max Temperature of Warmest Month

BIO6 Numeric. Min Temperature of Coldest Month

BIO7 Numeric. Temperature Annual Range

BIO8 Numeric. Mean Temperature of Wettest Quarter

BIO9 Numeric. Mean Temperature of Driest Quarter

BIO10 Numeric. Mean Temperature of Warmest Quarter

BIO11 Numeric. Mean Temperature of Coldest Quarter

BIO12 Numeric. Annual Precipitation

BIO13 Numeric. Precipitation of Wettest Month

BIO14 Numeric. Precipitation of Driest Month

BIO15 Numeric. Precipitation Seasonality

BIO16 Numeric. Precipitation of Wettest Quarter

BIO17 Numeric. Precipitation of Driest Quarter

BIO18 Numeric. Precipitation of Warmest Quarter

BIO19 Numeric. Precipitation of Coldest Quarter

Details

This dataset demonstrates a common problem in ecological modeling: bioclimatic predictors are highly correlated within groups (temperature variables BIO1-BIO11 are highly correlated; precipitation variables BIO12-BIO19 are moderately correlated), leading to multicollinearity issues. The species richness response depends on a subset of predictors.

Use case: Demonstrating `corrPrune()` and `modelPrune()` for reducing correlated environmental predictors before fitting species distribution models.

Source

Simulated data based on the 19 WorldClim bioclimatic variables

See Also

`corrPrune()`, `modelPrune()`

Examples

```
data(bioclim_example)

# The 19 WorldClim bioclimatic variables (https://www.worldclim.org/data/bioclim.html)
# Many are highly correlated, making them ideal for pruning

# Remove highly correlated variables
pruned <- corrPrune(bioclim_example[, -1], threshold = 0.7)
ncol(pruned)  # Reduced from 19 to ~8 variables

# Model-based pruning with VIF
model_data <- modelPrune(species_richness ~ .,
                         data = bioclim_example,
                         limit = 5)
attr(model_data, "selected_vars")
```

CorrCombo *CorrCombo Class*

Description

Holds the result of `corrSelect` or `MatSelect`: a list of valid variable combinations and their correlation statistics.

This class stores all subsets of variables that meet the specified correlation constraint, along with metadata such as the algorithm used, correlation method(s), variables forced into every subset, and summary statistics for each combination.

Usage

```
CorrCombo(
  subset_list = list(),
  avg_corr = numeric(0),
  min_corr = numeric(0),
  max_corr = numeric(0),
  var_names = character(0),
  threshold = numeric(0),
  forced_in = character(0),
  search_type = character(0),
  cor_method = character(0),
  n_rows_used = integer(0)
)

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

Arguments

`subset_list`	A list of character vectors. Each vector is a valid subset (variable names).
`avg_corr`	A numeric vector. Average absolute correlation within each subset.
`min_corr`	A numeric vector. Minimum pairwise absolute correlation in each subset.
`max_corr`	A numeric vector. Maximum pairwise absolute correlation within each subset.
`var_names`	Character vector of all variable names used for decoding.
`threshold`	Numeric scalar. The correlation threshold used during selection.
`forced_in`	Character vector. Variable names forced into each subset. Defaults to `character()`.
`search_type`	Character string. One of `"els"` or `"bron-kerbosch"`.
`cor_method`	Character string. Correlation method or `"mixed"`. Defaults to `character()`.
`n_rows_used`	Integer. Number of rows used for computing the correlation matrix.
`x`	A `CorrCombo` object to be printed.
`...`	Additional arguments (ignored).

Details

Properties:

subset_list A list of character vectors. Each vector is a valid subset (variable names).

avg_corr A numeric vector. Average absolute correlation within each subset.

min_corr A numeric vector. Minimum pairwise absolute correlation in each subset.

max_corr A numeric vector. Maximum pairwise absolute correlation within each subset.

var_names Character vector of all variable names used for decoding.

threshold Numeric scalar. The correlation threshold used during selection.

forced_in Character vector. Variable names that were forced into each subset.

search_type Character string. One of `"els"` or `"bron-kerbosch"`.

cor_method Character string. Either a single method (e.g. "pearson") or "mixed" if multiple methods used.

n_rows_used Integer. Number of rows used for computing the correlation matrix (after removing missing values).

See Also

`corrSelect`, `MatSelect`, `corrSubset`

Examples

```
print(CorrCombo(
  subset_list = list(c("A", "B"), c("A", "C")),
  avg_corr = c(0.2, 0.3),
  min_corr = c(0.1, 0.2),
  max_corr = c(0.3, 0.4),
  var_names = c("A", "B", "C"),
  threshold = 0.5,
  forced_in = character(),
  search_type = "els",
  cor_method = "mixed",
  n_rows_used = 5L
))
```

`corrPrune` *Association-Based Predictor Pruning*

Description

`corrPrune()` performs model-free variable subset selection by iteratively removing predictors until all pairwise associations fall below a specified threshold. It returns a single pruned data frame with predictors that satisfy the association constraint.

Usage

```
corrPrune(
  data,
  threshold = 0.7,
  measure = "auto",
  mode = "auto",
  force_in = NULL,
  by = NULL,
  group_q = 1,
  max_exact_p = 100,
  ...
)
```

Arguments

`data`	A data.frame containing candidate predictors.
`threshold`	Numeric scalar. Maximum allowed pairwise association (default: 0.7). Must be non-negative.
`measure`	Character string specifying the association measure to use. Options: `"auto"` (default), `"pearson"`, `"spearman"`, `"kendall"`, `"cramersv"`, `"eta"`, etc. When `"auto"`, Pearson correlation is used for all-numeric data, and appropriate measures are selected for mixed-type data.
`mode`	Character string specifying the search algorithm. Options: • `"auto"` (default): uses exact search if number of predictors <= `max_exact_p`, otherwise uses greedy search • `"exact"`: exhaustive search for maximal subsets (may be slow for large p) • `"greedy"`: fast approximate search using iterative removal
`force_in`	Character vector of variable names that must be retained in the final subset. Default: NULL.
`by`	Character vector naming one or more grouping variables. If provided, associations are computed separately within each group, then aggregated using the quantile specified by `group_q`. Default: NULL (no grouping).
`group_q`	Numeric scalar in (0, 1]. Quantile used to aggregate associations across groups when `by` is provided. Default: 1 (maximum, ensuring threshold holds in all groups). Use 0.9 for 90th percentile, etc.
`max_exact_p`	Integer. Maximum number of predictors for which exact mode is used when `mode = "auto"`. Default: 100.
`...`	Additional arguments (reserved for future use).

Details

`corrPrune()` identifies a subset of predictors whose pairwise associations are all below `threshold`. The function works in several stages:

1. **Variable type detection**: Identifies numeric vs. categorical predictors

2. **Association measurement**: Computes appropriate pairwise associations
3. **Grouping (optional)**: If by is specified, computes associations within each group and aggregates using the specified quantile
4. **Feasibility check**: Verifies that `force_in` variables satisfy the threshold constraint
5. **Subset selection**: Uses either exact or greedy search to find a valid subset

Grouped Pruning: When by is provided, the function ensures the selected predictors satisfy the threshold constraint across groups. For example, with `group_q = 1` (default), the returned predictors will have pairwise associations below `threshold` in *all* groups. With `group_q = 0.9`, they will satisfy the constraint in at least 90% of groups.

Mode Selection: Exact mode guarantees finding all maximal subsets and returns the largest one. Greedy mode is faster but approximate, using an iterative removal strategy based on association scores.

Tie-Breaking: When multiple subsets or variables are equally good, deterministic tie-breaking is applied:

- **Exact mode**: Selects by (1) largest subset size, (2) lowest average correlation, (3) alphabetically first variable names. Column order does not affect the result.
- **Greedy mode**: Removes the variable with (1) most constraint violations, (2) highest max association, (3) highest average association, (4) lowest column index. Column order can influence the result when earlier criteria are tied.

To see all maximal subsets instead of a single selection, use `corrSelect()`.

Value

A data.frame containing the pruned subset of predictors. The result has the following attributes:

selected_vars Character vector of retained variable names

removed_vars Character vector of removed variable names

mode Character string indicating which mode was used ("exact" or "greedy")

measure Character string indicating which association measure was used

threshold The threshold value used

See Also

`corrSelect` for exhaustive subset enumeration, `assocSelect` for mixed-type data subset enumeration, `modelPrune` for model-based predictor pruning.

Examples

```
# Basic numeric data pruning
data(mtcars)
pruned <- corrPrune(mtcars, threshold = 0.7)
names(pruned)

# Force certain variables to be included
pruned <- corrPrune(mtcars, threshold = 0.7, force_in = "mpg")
```

```
# Use greedy mode for faster computation
pruned <- corrPrune(mtcars, threshold = 0.7, mode = "greedy")
```

`corrSelect` *Select Variable Subsets with Low Correlation (Data Frame Interface)*

Description

Identifies combinations of numeric variables in a data frame such that all pairwise absolute correlations fall below a specified threshold. This function is a wrapper around `MatSelect()` and accepts data frames, tibbles, or data tables with automatic preprocessing.

Usage

```
corrSelect(
  df,
  threshold = 0.7,
  method = NULL,
  force_in = NULL,
 cor_method = c("pearson", "spearman", "kendall", "bicor", "distance", "maximal"),
  ...
)
```

Arguments

`df`	A data frame. Only numeric columns are used.
`threshold`	A numeric value in (0, 1). Maximum allowed absolute correlation. Defaults to 0.7.
`method`	Character. Selection algorithm to use. One of `"els"` or `"bron-kerbosch"`. If not specified, the function chooses automatically: `"els"` when `force_in` is provided, otherwise `"bron-kerbosch"`.
`force_in`	Optional character vector or numeric indices of columns to force into all subsets.
`cor_method`	Character string indicating which correlation method to use. One of `"pearson"` (default), `"spearman"`, `"kendall"`, `"bicor"`, `"distance"`, or `"maximal"`.
`...`	Additional arguments passed to `MatSelect()`, e.g., `use_pivot`.

Details

Only numeric columns are used for correlation analysis. Non-numeric columns (factors, characters, logicals, etc.) are ignored, and their names and types are printed to inform the user. These can be optionally reattached later using `corrSubset()` with `keepExtra = TRUE`.

Rows with missing values are removed before computing correlations. A warning is issued if any rows are dropped.

The `cor_method` controls how the correlation matrix is computed:

- "pearson": Standard linear correlation.
- "spearman": Rank-based monotonic correlation.
- "kendall": Kendall's tau.
- "bicor": Biweight midcorrelation (WGCNA::bicor).
- "distance": Distance correlation (energy::dcor).
- "maximal": Maximal information coefficient (minerva::mine).

For "bicor", "distance", and "maximal", the corresponding package must be installed.

Value

An object of class CorrCombo, containing selected subsets and correlation statistics.

See Also

assocSelect(), MatSelect(), corrSubset()

Examples

```
set.seed(42)
n <- 100

# Create 20 variables: 5 blocks of correlated variables + some noise
block1 <- matrix(rnorm(n * 4), ncol = 4)
block2 <- matrix(rnorm(n), ncol = 1)
block2 <- matrix(rep(block2, 4), ncol = 4) + matrix(rnorm(n * 4, sd = 0.1), ncol = 4)
block3 <- matrix(rnorm(n * 4), ncol = 4)
block4 <- matrix(rnorm(n * 4), ncol = 4)
block5 <- matrix(rnorm(n * 4), ncol = 4)

df <- as.data.frame(cbind(block1, block2, block3, block4, block5))
colnames(df) <- paste0("V", 1:20)

# Add a non-numeric column to be ignored
df$label <- factor(sample(c("A", "B"), n, replace = TRUE))

# Basic usage
corrSelect(df, threshold = 0.8)

# Try Bron-Kerbosch with pivoting
corrSelect(df, threshold = 0.6, method = "bron-kerbosch", use_pivot = TRUE)

# Force in a specific variable and use Spearman correlation
corrSelect(df, threshold = 0.6, force_in = "V10", cor_method = "spearman")
```

corrSubset *Extract Variable Subsets from a CorrCombo Object*

Description

Extracts one or more variable subsets from a CorrCombo object as data frames. Typically used after corrSelect or MatSelect to obtain filtered versions of the original dataset containing only low-correlation variable combinations.

Usage

```
corrSubset(res, df, which = "best", keepExtra = FALSE)
```

Arguments

res	A CorrCombo object returned by corrSelect or MatSelect.
df	A data frame or matrix. Must contain all variables listed in res@var_names. Columns not in res@var_names are ignored unless keepExtra = TRUE.
which	Subsets to extract. One of: • "best" (default) or 1: the top-ranked subset. • A single integer (e.g. 2): the nth ranked subset. • A vector of integers (e.g. 1:3): multiple subsets. • "all": all available subsets. Subsets are ranked by decreasing size, then increasing average correlation.
keepExtra	Logical. If TRUE, columns in df not in res@var_names (e.g., factors, characters) are retained. Defaults to FALSE.

Value

A data frame if a single subset is extracted, or a list of data frames if multiple subsets are extracted. Each data frame contains the selected variables (and optionally extras).

Note

A warning is issued if any rows contain missing values in the selected variables.

See Also

corrSelect, MatSelect, CorrCombo

Examples

```
# Simulate input data
set.seed(123)
df <- as.data.frame(matrix(rnorm(100), nrow = 10))
colnames(df) <- paste0("V", 1:10)

# Compute correlation matrix
cmat <- cor(df)

# Select subsets using corrSelect
res <- corrSelect(cmat, threshold = 0.5)

# Extract the best subset (default)
corrSubset(res, df)

# Extract the second-best subset
corrSubset(res, df, which = 2)

# Extract the first three subsets
corrSubset(res, df, which = 1:3)

# Extract all subsets
corrSubset(res, df, which = "all")

# Extract best subset and retain additional numeric column
df$CopyV1 <- df$V1
corrSubset(res, df, which = 1, keepExtra = TRUE)
```

---

cor_example *Example Correlation Matrix with Block Structure*

Description

A 20x20 correlation matrix with known block structure designed for demonstrating threshold selection, algorithm comparison, and visualization examples in vignettes.

Usage

```
cor_example
```

Format

A 20x20 numeric correlation matrix with row and column names V1-V20. The matrix has four distinct correlation blocks:

Block 1 (V1-V5) High correlation: mean = 0.81, range = (0.75, 0.95)

Block 2 (V6-V10) Moderate correlation: mean = 0.57, range = (0.5, 0.7)

Block 3 (V11-V15) Low correlation: mean = 0.28, range = (0.2, 0.4)

Block 4 (V16-V20) Minimal correlation: mean = 0.06, range = (0.0, 0.15)

Between-block correlations are low (range = (0.0, 0.3)). The matrix is guaranteed to be positive definite.

Details

This dataset provides a controlled correlation structure useful for:

- Threshold sensitivity analysis (comparing results at tau = 0.5, 0.7, 0.9)
- Algorithm comparison (exact vs greedy modes)
- Visualization examples (heatmaps, correlation distributions)
- Reproducible benchmarks across vignettes

Expected behavior with different thresholds:

- tau = 0.5: Block 1 requires pruning (all pairs > 0.75)
- tau = 0.7: Blocks 1-2 require pruning
- tau = 0.9: Only Block 1 requires pruning

Source

Generated with `data-raw/create_cor_example.R` using seed 20250125.

Examples

```
data(cor_example)

# Matrix dimensions
dim(cor_example)

# Visualize structure
if (requireNamespace("corrplot", quietly = TRUE)) {
  corrplot::corrplot(cor_example, method = "color", type = "upper",
                     tl.col = "black", tl.cex = 0.7)
}

# Distribution of correlations
hist(cor_example[upper.tri(cor_example)],
     breaks = 30,
     main = "Distribution of Correlations in cor_example",
     xlab = "Correlation",
     col = "steelblue")

# Use with MatSelect
library(corrselect)
results <- MatSelect(cor_example, threshold = 0.7, method = "els")
show(results)
```

genes_example *Example Gene Expression Data for Bioinformatics*

Description

A simulated gene expression dataset with 200 genes measured across 100 samples, organized into co-expression modules with a binary disease outcome.

Usage

```
genes_example
```

Format

A data frame with 100 rows and 202 variables:

sample_id Character. Unique sample identifier

disease_status Factor. Disease status (Healthy, Disease)

GENE001, GENE002, GENE003, GENE004, GENE005, GENE006, GENE007, GENE008, GENE009, GENE010, GE Numeric. Gene expression values (log-transformed)

Details

This dataset simulates a high-dimensional, low-sample scenario common in genomics. Genes are organized into four co-expression modules:

- Module 1 (GENE001-GENE050): Highly correlated (r ~= 0.80), disease-associated
- Module 2 (GENE051-GENE100): Moderately correlated (r ~= 0.60)
- Module 3 (GENE101-GENE150): Weakly correlated (r ~= 0.40)
- Module 4 (GENE151-GENE200): Independent (r ~= 0)

Disease outcome depends on a subset of genes from Module 1.

Use case: Demonstrating `corrPrune()` with `mode = "greedy"` for handling high-dimensional data efficiently.

Source

Simulated data based on typical gene expression microarray structures

See Also

`corrPrune()`

Examples

```
data(genes_example)

# Greedy pruning for high-dimensional data
gene_data <- genes_example[, -(1:2)]  # Exclude ID and outcome
pruned <- corrPrune(gene_data, threshold = 0.8, mode = "greedy")
ncol(pruned)  # Reduced from 200 to ~50 genes

# Use pruned genes for classification
pruned_with_outcome <- data.frame(
  disease_status = genes_example$disease_status,
  pruned
)
```

---

longitudinal_example *Example Longitudinal Data for Clinical Research*

Description

A simulated longitudinal study dataset with 50 subjects measured at 10 timepoints each, with 20 correlated predictors and nested random effects (subject and site).

Usage

```
longitudinal_example
```

Format

A data frame with 500 rows and 25 variables:

obs_id Integer. Observation identifier (1-500)

subject Factor. Subject identifier (1-50)

site Factor. Study site identifier (1-5)

time Integer. Measurement timepoint (1-10)

outcome Numeric. Continuous outcome variable

x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20 Numeric. Correlated predictor variables

Details

This dataset represents a typical longitudinal study with repeated measures. Predictors are correlated both within and between subjects:

- Predictors x1-x10: Highly correlated (r ~= 0.75)
- Predictors x11-x20: Moderately correlated (r ~= 0.50)

The outcome depends on time (linear trend), random effects (subject and site), and a subset of fixed-effect predictors (x1, x5, x15).

Use case: Demonstrating `modelPrune()` with mixed models (`lme4` engine) to prune fixed effects while preserving random effects structure.

Source

Simulated data based on typical clinical trial designs

See Also

`modelPrune()`

Examples

```
data(longitudinal_example)

## Not run:
# Prune fixed effects in mixed model (requires lme4)
if (requireNamespace("lme4", quietly = TRUE)) {
  pruned <- modelPrune(
    outcome ~ x1 + x2 + x3 + x4 + x5 + (1|subject) + (1|site),
    data = longitudinal_example,
    engine = "lme4",
    limit = 5
  )

  # Random effects preserved, only fixed effects pruned
  attr(pruned, "selected_vars")
}

## End(Not run)
```

MatSelect *Select Variable Subsets with Low Correlation or Association (Matrix Interface)*

Description

Identifies all maximal subsets of variables from a symmetric matrix (typically a correlation matrix) such that all pairwise absolute values stay below a specified threshold. Implements exact algorithms such as Eppstein–Löffler–Strash (ELS) and Bron–Kerbosch (with or without pivoting).

Usage

```
MatSelect(mat, threshold = 0.7, method = NULL, force_in = NULL, ...)
```

Arguments

mat	A numeric, symmetric matrix with 1s on the diagonal (e.g. correlation matrix). Column names (if present) are used to label output variables.
threshold	A numeric scalar in (0, 1). Maximum allowed absolute pairwise value. Defaults to 0.7.
method	Character. Selection algorithm to use. One of "els" or "bron-kerbosch". If not specified, the function chooses automatically: "els" when force_in is provided, otherwise "bron-kerbosch".
force_in	Optional integer vector of 1-based column indices to force into every subset.
...	Additional arguments passed to the backend, e.g., use_pivot (logical) for enabling pivoting in Bron–Kerbosch (ignored by ELS).

Value

An object of class CorrCombo, containing all valid subsets and their correlation statistics.

Examples

```
set.seed(42)
mat <- matrix(rnorm(100), ncol = 10)
colnames(mat) <- paste0("V", 1:10)
cmat <- cor(mat)

# Default method (Bron-Kerbosch)
res1 <- MatSelect(cmat, threshold = 0.5)

# Bron-Kerbosch without pivot
res2 <- MatSelect(cmat, threshold = 0.5, method = "bron-kerbosch", use_pivot = FALSE)

# Bron-Kerbosch with pivoting
res3 <- MatSelect(cmat, threshold = 0.5, method = "bron-kerbosch", use_pivot = TRUE)

# Force variable 1 into every subset (with warning if too correlated)
res4 <- MatSelect(cmat, threshold = 0.5, force_in = 1)
```

---

modelPrune *Model-Based Predictor Pruning*

Description

modelPrune() performs iterative removal of fixed-effect predictors based on model diagnostics (e.g., VIF) until all remaining predictors satisfy a specified threshold. It supports linear models, generalized linear models, and mixed models.

Usage

```
modelPrune(
  formula,
  data,
  engine = "lm",
  criterion = "vif",
  limit = 5,
  force_in = NULL,
  max_steps = NULL,
  ...
)
```

Arguments

formula	A model formula specifying the response and predictors. May include random effects for mixed models (e.g., y ~ x1 + x2 + (1|group)).
data	A data.frame containing the variables in the formula.
engine	Either a character string for built-in engines, or a list defining a custom engine. **Built-in engines** (character string): • "lm" (default): Linear models via stats::lm() • "glm": Generalized linear models via stats::glm() (requires family argument) • "lme4": Mixed models via lme4::lmer() or lme4::glmer() (requires lme4 package) • "glmmTMB": Generalized linear mixed models via glmmTMB::glmmTMB() (requires glmmTMB package) **Custom engine** (named list with required components): • fit: function(formula, data, ...) that returns a fitted model object • diagnostics: function(model, fixed_effects) that returns a named numeric vector of diagnostic scores (one per fixed effect, higher values = worse) • name (optional): character string used in error messages (default: "custom")
criterion	Character string specifying the diagnostic criterion for pruning. For built-in engines, supported values are: • "vif" (default): Variance Inflation Factor. Measures how much the variance of a coefficient is inflated due to collinearity. Values > 5-10 indicate problematic multicollinearity. • "condition_number": Condition indices based on singular value decomposition of the design matrix. Higher values indicate greater collinearity. For custom engines, this parameter is ignored (diagnostics are computed by the engine's diagnostics function).
limit	Numeric scalar. Maximum allowed value for the criterion. Predictors with diagnostic values exceeding this limit are iteratively removed. Default: 5 (common VIF threshold).
force_in	Character vector of predictor names that must be retained in the final model. These variables will not be removed during pruning. Default: NULL.

`max_steps` Integer. Maximum number of pruning iterations. If NULL (default), pruning continues until all diagnostics are below the limit or no more removable predictors remain.

`...` Additional arguments passed to the modeling function (e.g., `family` for glm/glmer, control parameters for lme4/glmmTMB).

Details

`modelPrune()` works by:

1. Parsing the formula to identify fixed-effect predictors
2. Fitting the initial model
3. Computing diagnostics for each fixed-effect predictor
4. Checking feasibility of `force_in` constraints
5. Iteratively removing the predictor with the worst diagnostic value (excluding `force_in` variables) until all diagnostics <= `limit`
6. Returning the pruned data frame

Random Effects: For mixed models (lme4, glmmTMB), only fixed-effect predictors are considered for pruning. Random-effect structure is preserved exactly as specified in the original formula.

VIF Computation: Variance Inflation Factors are computed from the fixed-effects design matrix. For categorical predictors, VIF represents the inflation for the entire factor (not individual dummy variables).

Determinism: The algorithm is deterministic. Ties in diagnostic values are broken by removing the predictor that appears last in the formula.

Force-in Constraints: If variables in `force_in` violate the diagnostic threshold, the function will error. This ensures that the constraint is feasible before pruning begins.

Value

A data.frame containing only the retained predictors (and response). The result has the following attributes:

selected_vars Character vector of retained predictor names

removed_vars Character vector of removed predictor names (in order of removal)

engine Character string indicating which engine was used (for custom engines, this is the engine's `name` field)

criterion Character string indicating which criterion was used

limit The threshold value used

final_model The final fitted model object (optional)

See Also

`corrPrune` for association-based predictor pruning, `corrSelect` for exhaustive subset enumeration.

Examples

```
# Linear model with VIF-based pruning
data(mtcars)
pruned <- modelPrune(mpg ~ ., data = mtcars, engine = "lm", limit = 5)
names(pruned)

# Force certain predictors to remain
pruned <- modelPrune(mpg ~ ., data = mtcars, force_in = "drat", limit = 20)

# GLM example (requires family argument)
pruned <- modelPrune(am ~ ., data = mtcars, engine = "glm",
                     family = binomial(), limit = 5)

## Not run:
# Custom engine example (INLA)
inla_engine <- list(
  name = "inla",
  fit = function(formula, data, ...) {
    inla::inla(formula = formula, data = data,
               family = list(...)$family %||% "gaussian",
               control.compute = list(config = TRUE))
  },
  diagnostics = function(model, fixed_effects) {
    scores <- model$summary.fixed[, "sd"]
    names(scores) <- rownames(model$summary.fixed)
    scores[fixed_effects]
  }
)

pruned <- modelPrune(y ~ x1 + x2 + x3, data = df,
                     engine = inla_engine, limit = 0.5)

## End(Not run)
```

---

survey_example *Example Survey Data for Social Science Research*

Description

A simulated questionnaire dataset with 30 Likert-scale items measuring three latent constructs (satisfaction, engagement, loyalty), plus demographic variables and an overall satisfaction score.

Usage

```
survey_example
```

Format

A data frame with 200 rows and 35 variables:

respondent_id Integer. Unique respondent identifier

age Integer. Respondent age (18-75 years)

gender Factor. Gender (Male, Female, Other)

education Ordered factor. Education level (High School, Bachelor, Master, PhD)

overall_satisfaction Integer. Overall satisfaction score (0-100)

satisfaction_1, satisfaction_2, satisfaction_3, satisfaction_4, satisfaction_5, satisfaction_6, satisfaction_7, satisfaction_
Ordered factor. Satisfaction items (1-7 Likert scale)

engagement_1, engagement_2, engagement_3, engagement_4, engagement_5, engagement_6, engagement_7, engager
Ordered factor. Engagement items (1-7 Likert scale)

loyalty_1, loyalty_2, loyalty_3, loyalty_4, loyalty_5, loyalty_6, loyalty_7, loyalty_8, loyalty_9, loyalty_10
Ordered factor. Loyalty items (1-7 Likert scale)

Details

This dataset represents a common scenario in survey research: multiple items measuring similar constructs lead to redundancy and multicollinearity. Items within each construct are correlated (satisfaction, engagement, loyalty), and the constructs themselves are inter-correlated.

Use case: Demonstrating `assocSelect()` for identifying redundant questionnaire items in mixed-type data (ordered factors + numeric variables).

Source

Simulated data based on typical customer satisfaction survey structures

See Also

`assocSelect()`, `corrPrune()`

Examples

```
data(survey_example)

# This dataset has mixed types: numeric (age, overall_satisfaction),
# factors (gender, education), and ordered factors (Likert items)
str(survey_example[, 1:10])


# Use assocSelect() for mixed-type data pruning
# This may take a few seconds with 34 variables
pruned <- assocSelect(survey_example[, -1],  # Exclude respondent_id
                      threshold = 0.8,
                      method_ord_ord = "spearman")
length(attr(pruned, "selected_vars"))
```

Index