

Package: corila (via r-universe)

October 11, 2026

Version 0.1.0

Title Sparse modelling with grouped and correlated features allowing for privileged information

Description Implements lasso regression (Gaussian, binomial, Poisson, and Cox) for features in heterogeneous groups (e.g., where some groups contain subgroups of correlated features), overlapping groups (i.e., where some features belong to two or more groups), and misspecified groups (i.e., where some group labels are incorrect or missing). Involves an initial ridge regression followed by a final adaptive lasso regression. Exchanges information between features according to their assumed adjacencies or their observed correlations to improve selection and predictive performance. Allows for features that can provide information on other features but must not themselves be selected (i.e., privileged information).

Depends R (>= 4.0.0)

Imports stats, checkmate, glmnet, multiridge

Suggests MASS, mvtnorm, pROC, survival, knitr, testthat, remotes, rmarkdown, spelling

Encoding UTF-8

VignetteBuilder knitr

Language en-GB

URL <https://github.com/rauschenberger/corila>,
<https://rauschenberger.github.io/corila/>

BugReports <https://github.com/rauschenberger/corila/issues>

License GPL (>= 3)

Config/roxygen2/version 8.1.0

Roxygen list(markdown = TRUE, roclets = c("`namespace", "`rd",
 "`srr::srr_stats_roclet"))

Repository <https://ropensci-staging.r-universe.dev>

Date/Publication 2026-10-09 14:22:40 UTC
RemoteUrl <https://github.com/rauschenberger/corila>
RemoteRef HEAD
RemoteSha 1be89d13dee4d79b96c3b9e73886dc6cdf26438

Contents

corila-package	2
coef.cv.corila	3
coef.multiridge	4
cv.corila	5
deviance.cv.corila	8
fitted.cv.corila	9
methods	10
multiridge	11
nobs.cv.corila	13
plot.cv.corila	14
predict.cv.corila	15
predict.multiridge	16
residuals.cv.corila	17
simulate_data	18
Index	21

corila-package	<i>Sparse modelling with grouped and correlated features allowing for privileged information</i>
----------------	--

Description

The R package `corila` implements "Sparse modelling with grouped and correlated features allowing for privileged information" (*Rauschenberger, 2026*). This is the first implementation of a novel algorithm, sharing information based on assumed adjacencies and observed correlations, and allowing not only for primary but also for auxiliary predictors. It builds upon adaptive lasso regression with the [glmnet-package](#).

Details

Use function `cv.corila()` for model fitting. Type `library(corila)` and then `?cv.corila` or `help("cv.corila")` to open its help file.

See the vignette for further examples. Type `vignette("corila")` or `browseVignettes("corila")` to open the vignette.

This package also includes the wrapper function `multiridge()` for multi-penalty ridge regression with the [multiridge-package](#).

Author(s)

Armin Rauschenberger

References

Armin Rauschenberger (2026). "Sparse modelling with grouped and correlated features allowing for privileged information". *In preparation*.

See Also

First use `cv.corila()` to fit the model, and then `coef()` to extract coefficients or `predict()` to make predictions.

Examples

```
?cv.corila
?coef.cv.corila
?predict.cv.corila
```

coef.cv.corila	<i>Extract coefficients</i>
----------------	-----------------------------

Description

[Stable]

Extracts coefficients from an object of class "cv.corila".

Usage

```
## S3 method for class 'cv.corila'
coef(object, s = "lambda.min", ...)
```

Arguments

object	object of class "cv.corila"
s	character "lambda.min" or numeric non-negative scalar
...	(for compatibility with <code>stats::coef</code>)

Details

This function calls `.combine_slopes()` to combine positive and negative coefficients and `.backscale()` to bring coefficients back to the original scale.

Value

Returns an $(1 + p)$ -dimensional vector of the estimated coefficients. The first entry is the estimated intercept, and the other p entries are the estimated slopes.

References

[Armin Rauschenberger](#) (2026). "Sparse modelling with grouped and correlated features allowing for privileged information". *In preparation*.

See Also

Fit models with `cv.corila()` and make predictions with `predict()`.

Examples

```
data <- simulate_data()
model <- cv.corila(x = data$x_train,
                  y = data$y_train,
                  group = data$group,
                  primary = data$primary)
beta_hat <- coef(object = model)
y_hat <- predict(object = model, newx = data$x_test)

# example for automatic mutation testing (with the R package autotest)
data <- simulate_data()
model <- cv.corila(x = data$x_train,
                  y = data$y_train,
                  group = as.double(data$group),
                  primary = data$primary,
                  alpha_init = 0.0,
                  foldid = rep(x = seq_len(10L),
                              length.out = nrow(data$x_train)))
```

coef.multiridge	<i>Extract coefficients</i>
-----------------	-----------------------------

Description

[Stable]

Extracts coefficients from a multi-penalty ridge regression model.

Usage

```
## S3 method for class 'multiridge'
coef(object, ...)
```

Arguments

object	object of type "multiridge"
...	(for compatibility with <code>stats::coef</code>)

Value

Returns an $(1 + p)$ -dimensional vector of estimated coefficients (estimated intercept and estimated slopes) if family="gaussian" and family="binomial" and a p -dimensional vector of estimated slopes if family="cox".

References

Mark A. van de Wiel, Mirrelijn M. van Nee and Armin Rauschenberger (2021). "Fast cross-validation for multi-penalty high-dimensional ridge regression" *Journal of Computational and Graphical Statistics* 30(4):835-847. doi:10.1080/10618600.2021.1904962.

See Also

Fit models with `multiridge()` and make predictions with `predict()`.

Examples

```
data <- simulate_data(prob_primary = 1.0)
model <- multiridge(x = data$x_train, y = data$y_train, group = data$group)
beta_hat <- coef(model)
y_hat <- predict(object = model, newx = data$x_test)
```

cv.corila

Sparse group lasso regression

Description**[Experimental]**

Optimises the parameters and the hyperparameters of the sparse group lasso.

Usage

```
cv.corila(
  x,
  y,
  group,
  primary = NULL,
  family = "gaussian",
  alpha_init = 0,
  cor = "pearson",
  alpha_final = 1,
  nfolds = 10L,
  foldid = NULL,
```

```

    tune = "weight",
    na_action = "error",
    silent = FALSE
)

```

Arguments

x	$n_0 \times p$ predictor matrix, containing only numerical values (continuous, integer, or binary), where n_0 is the number of observations used for model training and p is the number of predictors
y	response vector of length n_0 , containing numerical values (family="gaussian"), integer values (family="poisson"), binary values (family="binomial"), or a survival object created with <code>survival::Surv()</code> (family="cox"), where n_0 is the number of observations used for model training
group	group structure (multiple options): • p-dimensional vector of group indices (in $\{1, \dots, q\}$) or labels, • list with q slots containing the variable indices (in $\{1, \dots, p\}$) or labels, • $p \times p$ matrix, where the entry in the j^{th} row and the k^{th} column indicates whether information should be transferred from the j^{th} to the k^{th} variable (usually with a unit diagonal so that predictors retain the information from their initial coefficient)
primary	p -dimensional logical vector indicating whether a predictor may be included in the final model (TRUE for "primary predictors") or must be excluded from the final model (FALSE for "auxiliary predictors")
family	character string "gaussian", "binomial", "poisson", or "cox"
alpha_init	A scalar specifying the method used for obtaining initial coefficients: • a numeric scalar in the unit interval ($0 \leq \text{alpha_init} \leq 1$) to define the mixing parameter for elastic net regression (default: ridge penalisation with <code>alpha_init=0</code>); • the character scalar "pearson", "spearman", or "kendall" to use initial correlation coefficients (not implemented for family="cox") • the character scalar "multiridge" to use multi-penalty ridge regression with one penalty for each group (not implemented for family="poisson" or overlapping groups), • NA to set all initial coefficients equal to 1
cor	character string "pearson" (default), "spearman", or "kendall"; or a correlation matrix (p rows, p columns, entries between -1 and $+1$)
alpha_final	elastic net mixing parameter for final regression: numeric between 0 for ridge penalisation and 1 for lasso penalisation (default: lasso penalisation with <code>alpha_final=1</code>)
nfolds	positive integer specifying the number of folds (minimum 3, maximum n) NB: If <code>foldid</code> is provided, <code>nfolds</code> is overwritten by <code>max(foldid)</code> .
foldid	n_0 -dimensional vector containing the fold identifiers (minimum 1, maximum <code>nfolds</code>)
tune	character string for determining the candidate values for the hyperparameters:

- "none": fixed weights and exponents ($\text{wgt_local}=1$, $\text{exp_local}=0$, $\text{wgt_global}=0$), no tuning (share information within groups, ignore correlations)
- "weight" (default): fixed exponents ($\text{exp_local}=0$, $\text{exp_global}=2$), tuning $\text{wgt_local}=1-\text{wgt_global}$ (find a compromise between sharing information within groups and between correlated predictors)
- "exponent": fixed weights ($\text{wgt_local}=1$, $\text{wgt_global}=0$), tuning exp_local (share information between correlated predictors in the same group, determine level of trust in correlation coefficients)
- "bivariate": tuning $\text{wgt_local}=1-\text{wgt_global}$ and $\text{exp_local}=\text{exp_global}$ (find a compromise between sharing information between predictors in the same group and between all predictors, determine level of trust in correlation coefficients)
- "factorial": tuning $\text{wgt_local}=1-\text{wgt_global}$, exp_local , and exp_global (find a compromise between sharing information within groups and between correlated predictors, with possibly different exponents)

(The internal function `.set_candidates()` uses this argument to create a grid of candidates values for wgt_local , exp_local , wgt_global , and exp_global .)

<code>na_action</code>	character "error" to trigger an error if any observation has a missing predictor or a missing response or "complete_cases" to exclude observations with a missing predictor or a missing response from model fitting (while providing fitted values for these observations)
<code>silent</code>	Should messages from <code>glmnet::glmnet()</code> and <code>glmnet::cv.glmnet()</code> be suppressed? (logical scalar, FALSE or TRUE)

Details

The numbers of observations (samples) for training or testing are indicated by n_0 and n_1 , respectively, the number of predictors (features) is indicated by p , and the number of predictor group is indicated by q . Observations are indexed by i in $\{1, \dots, n\}$, predictors are indexed by j in $\{1, \dots, p\}$, and predictor groups are indexed by k in $\{1, \dots, q\}$. The number of predictors in the k^{th} group is indicated by p_k , with $\sum_{k=1}^q p_k = p$ for non-overlapping groups.

Value

Returns an object of class "cv.corila", a list with the following slots:

- `model`: list with one slot for each combination of hyperparameters, each slot contains an object of class "glmnet"
- `hyper`: data frame with one row for each combination of hyperparameters, four columns for the values of the hyperparameters (wgt_local , exp_local , wgt_global , and exp_global) and a column for the cross-validated loss (`cvm`)
- `id_hyper`: index of combination of hyperparameters leading to the lowest cross-validated loss
- `lambda.min` optimised regularisation hyperparameter
- `scale`: output from `.forescale()`
- `y_hat`: n_0 -dimensional vector of fitted values

References

[Armin Rauschenberger](#) (2026). "Sparse modelling with grouped and correlated features allowing for privileged information". *In preparation*.

See Also

Extract coefficients with [coef\(\)](#) and make predictions with [predict\(\)](#).

This user function repeatedly calls [corila\(\)](#) with different values for the regularisation and mixing hyperparameters.

The arguments of this function are validated with the helper functions listed [here](#).

Examples

```
data <- simulate_data()
model <- cv.corila(x = data$x_train,
                  y = data$y_train,
                  group = data$group,
                  primary = data$primary)
beta_hat <- coef(object = model)
y_hat <- predict(object = model, newx = data$x_test)

# example for automatic mutation testing (with the R package autotest)
data <- simulate_data()
model <- cv.corila(x = data$x_train,
                  y = data$y_train,
                  group = as.double(data$group),
                  primary = data$primary,
                  alpha_init = 0.0,
                  foldid = rep(x = seq_len(10L),
                              length.out = nrow(data$x_train)))
```

deviance.cv.corila	<i>Deviance</i>
--------------------	-----------------

Description

Calculates the deviance.

Usage

```
## S3 method for class 'cv.corila'
deviance(object, ...)
```

Arguments

object	object of class "cv.corila"
...	(for compatibility with stats::deviance)

Details

Returns the deviance calculated by `glmnet::deviance.glmnet()` for the model with the optimised mixing and regularisation hyperparameters.

Value

Returns a scalar.

See Also

The internal function `.deviance()` calculates the deviance from fitted and observed values.

Examples

```
# listing S3 methods
methods(class = "cv.corila")

# simulating data
n <- 10L; p <- 20L; q <- 5L
x <- matrix(rnorm(n * p), nrow = n , ncol = p)
y <- rnorm(n)
group <- rep(seq_len(q), length.out = p)
primary <- as.logical(rbinom(n = p, size = 1L, prob = 0.5))

# fitting the model
object <- cv.corila(x = x, y = y, group = group, primary = primary)

# using S3 methods
coef(object)
predict(object, newx = x)
fitted(object)
residuals(object)
plot(object)
print(object)
summary(object)
```

fitted.cv.corila	<i>Fitted values</i>
------------------	----------------------

Description

[Stable]

Extracts fitted values.

Usage

```
## S3 method for class 'cv.corila'
fitted(object, ...)
```

Arguments

object object of class "cv.corila"
 ... (for compatibility with `stats::fitted`)

Value

Returns a numeric vector of length n_0 (one entry for each training observation).

See Also

Use `predict()` to obtain predicted values (i.e., for testing observations).

Examples

```
# listing S3 methods
methods(class = "cv.corila")

# simulating data
n <- 10L; p <- 20L; q <- 5L
x <- matrix(rnorm(n * p), nrow = n , ncol = p)
y <- rnorm(n)
group <- rep(seq_len(q), length.out = p)
primary <- as.logical(rbinom(n = p, size = 1L, prob = 0.5))

# fitting the model
object <- cv.corila(x = x, y = y, group = group, primary = primary)

# using S3 methods
coef(object)
predict(object, newx = x)
fitted(object)
residuals(object)
plot(object)
print(object)
summary(object)
```

methods	<i>List of methods for class "cv.corila"</i>
---------	--

Description

Implemented S3 methods for objects of class "cv.corila":

- `coef()`: extracts estimated coefficients
- `predict()`: calculates predicted values
- `fitted()`: extracts fitted values
- `residuals()`: calculates deviance residuals

- `plot()`: visualises observed vs fitted values and estimated coefficients
- `print()`: prints information to the console
- `summary()`: summarises the fitted model
- `deviance()`: extracts the deviance
- `nobs()`: extracts the number of observations

Value

`coef()` returns a $(1+)$ p -dimensional vector, `predict()` returns an n_1 -dimensional vector, `fitted()` and `residuals()` return an n_0 -dimensional vector. See individual methods for details.

See Also

Use `cv.corila()` to fit the model.

Examples

```
# listing S3 methods
methods(class = "cv.corila")

# simulating data
n <- 10L; p <- 20L; q <- 5L
x <- matrix(rnorm(n * p), nrow = n , ncol = p)
y <- rnorm(n)
group <- rep(seq_len(q), length.out = p)
primary <- as.logical(rbinom(n = p, size = 1L, prob = 0.5))

# fitting the model
object <- cv.corila(x = x, y = y, group = group, primary = primary)

# using S3 methods
coef(object)
predict(object, newx = x)
fitted(object)
residuals(object)
plot(object)
print(object)
summary(object)
```

Description

[Experimental]

Fits multi-penalty ridge regression (tuning regularisation hyperparameters and estimating regression coefficients). This is a wrapper function of some functions from the [multiridge-package](#).

Usage

```
multiridge(
  x,
  y,
  group,
  family = "gaussian",
  foldid = NULL,
  nfolds = 10L,
  penalties = NULL
)
```

Arguments

x	predictors: $n_0 \times p$ numeric matrix
y	response: n_0 -dimensional vector
group	p -dimensional integer vector with entries in $\{1, \dots, q\}$
family	character "gaussian" (not "linear"), "binomial" (not "logistic"), or "cox"
foldid	n_0 -dimensional vector containing the fold identifiers (minimum 1, maximum nfolds)
nfolds	positive integer specifying the number of folds (minimum 3, maximum n) NB: If foldid is provided, nfolds is overwritten by <code>max(foldid)</code> .
penalties	q -dimensional vector of non-negative penalty parameters, or NULL (cross-validation)

Details

The numbers of observations (samples) for training or testing are indicated by n_0 and n_1 , respectively, the number of predictors (features) is indicated by p , and the number of predictor group is indicated by q . Observations are indexed by i in $\{1, \dots, n\}$, predictors are indexed by j in $\{1, \dots, p\}$, and predictor groups are indexed by k in $\{1, \dots, q\}$. The number of predictors in the k^{th} group is indicated by p_k , with $\sum_{k=1}^q p_k = p$ for non-overlapping groups.

Value

Returns an object of class "multiridge", a list with the following slots:

- slots from [IWLSridge\(\)](#) or [IWLScoxridge\(\)](#)
- character family with value "gaussian" (also for "linear"), "binomial" (also for "logistic"), or "cox"
- q -dimensional vector penalties containing optimised regularisation hyperparameters (one for each predictor group)
- list indices with nfolds slots (one for each cross-validation fold), each containing the indices of the observations
- list datablocks with q slots (one for each predictor group), each containing an $n_0 \times p_k$ matrix, where $k \in \{1, \dots, q\}$
- p -dimensional group vector group (see argument)
- list pars with slots family (see above), the p -dimensional vectors `mu.x` and `sd.x` and the scalars `mu.y` and `sd.y`

References

Mark A. van de Wiel, Mirrelijn M. van Nee and Armin Rauschenberger (2021). "Fast cross-validation for multi-penalty high-dimensional ridge regression" *Journal of Computational and Graphical Statistics* 30(4):835-847. doi:10.1080/10618600.2021.1904962.

See Also

Extract coefficients with `coef()` or make predictions with `predict()`. Use `cv.corila()` to estimate sparse models.

This wrapper function calls various functions from the `multiridge-package`, namely `createXXblocks()`, `fastCV2()`, `CVfolds()`, `optLambdasWrap()`, `SigmaFromBlocks()`, `IWLSridge()`, and `IWLScoxridge()`.

The `multiridge-package` accepts not only an $n \times p$ matrix but also a list of length q of $n \times p_k$ matrices, with k in $\{1, \dots, q\}$.

Examples

```
data <- simulate_data(prob_primary = 1.0)
model <- multiridge(x = data$x_train, y = data$y_train, group = data$group)
beta_hat <- coef(model)
y_hat <- predict(object = model, newx = data$x_test)
```

nobs.cv.corila	Observation Count
----------------	-------------------

Description

Extracts the number of observations.

Usage

```
## S3 method for class 'cv.corila'
nobs(object, ...)
```

Arguments

object	object of class "cv.corila"
...	(for compatibility with <code>stats::nobs</code>)

Value

Returns a positive integer.

Examples

```
# listing S3 methods
methods(class = "cv.corila")

# simulating data
n <- 10L; p <- 20L; q <- 5L
x <- matrix(rnorm(n * p), nrow = n , ncol = p)
y <- rnorm(n)
group <- rep(seq_len(q), length.out = p)
primary <- as.logical(rbinom(n = p, size = 1L, prob = 0.5))

# fitting the model
object <- cv.corila(x = x, y = y, group = group, primary = primary)

# using S3 methods
coef(object)
predict(object, newx = x)
fitted(object)
residuals(object)
plot(object)
print(object)
summary(object)
```

plot.cv.corila

Plot Sparse Group Lasso (S3 method)

Description

Plot method for class "cv.corila".

Usage

```
## S3 method for class 'cv.corila'
plot(x, ...)
```

Arguments

x	object of class "cv.corila"
...	(for compatibility with base::plot)

Details

This function generates two figures:

- a scatter plot of fitted versus observed values for the Gaussian and the Poisson families, a box plot of predicted probabilities for the two classes for the binomial family, or a histogram of fitted relative risks for the Cox model
- estimated coefficients versus indices of predictors

Value

Returns NULL invisibly.

Examples

```
n <- 12L # decrease to 10 to check LOOCV
p <- 20L
q <- 5L
x <- matrix(rnorm(n * p), nrow = n, ncol = p)
y <- rnorm(n)
group <- rep(seq_len(q), length.out = p)
primary <- as.logical(rbinom(n = p, size = 1L, prob = 0.5))
object <- cv.corila(x = x, y = y, group = group, primary = primary)
print(object)
summary(object)
```

predict.cv.corila	<i>predict (S3 method)</i>
-------------------	----------------------------

Description

[Stable]

Makes predictions from an object of class "cv.corila".

Usage

```
## S3 method for class 'cv.corila'
predict(object, newx, s = "lambda.min", ...)
```

Arguments

object	object of class "cv.corila"
newx	$n_0 \times p$ predictor matrix (training data) to obtain fitted values, $n_1 \times p$ predictor matrix (testing data) to obtain predicted values
s	character "lambda.min" or numeric non-negative scalar
...	(for compatibility with stats::predict)

Details

This function calls [.expand_auxiliary\(\)](#) for handling auxiliary predictors, [.forescale\(\)](#) for standardising the predictor matrix, and [.backscale\(\)](#) for bringing predicted values back to the original scale (if family="gaussian").

Value

Returns fitted or predicted values in an $n_0 \times m$ -dimensional or $n_1 \times m$ -dimensional matrix, respectively.

References

[Armin Rauschenberger](#) (2026). "Sparse modelling with grouped and correlated features allowing for privileged information". *In preparation*.

See Also

Fit models with `cv.corila()`, extract coefficients with `coef()`, and extract fitted values with `fitted()`.

Examples

```
data <- simulate_data()
model <- cv.corila(x = data$x_train,
                  y = data$y_train,
                  group = data$group,
                  primary = data$primary)
beta_hat <- coef(object = model)
y_hat <- predict(object = model, newx = data$x_test)

# example for automatic mutation testing (with the R package autotest)
data <- simulate_data()
model <- cv.corila(x = data$x_train,
                  y = data$y_train,
                  group = as.double(data$group),
                  primary = data$primary,
                  alpha_init = 0.0,
                  foldid = rep(x = seq_len(10L),
                              length.out = nrow(data$x_train)))
```

predict.multiridge	<i>Make predictions</i>
--------------------	-------------------------

Description

[Stable]

Makes predictions from a multi-penalty ridge regression model.

Usage

```
## S3 method for class 'multiridge'
predict(object, newx, ...)
```

Arguments

object	object of type "multiridge"
newx	$n_0 \times p$ predictor matrix (training data) to obtain fitted values, $n_1 \times p$ predictor matrix (testing data) to obtain predicted values
...	(for compatibility with <code>stats::predict</code>)

Value

Returns an n_0 -dimensional vector of fitted values or an n_1 -dimensional vector of predicted values.

References

Mark A. van de Wiel, Mirrelijm M. van Nee and Armin Rauschenberger (2021). "Fast cross-validation for multi-penalty high-dimensional ridge regression" *Journal of Computational and Graphical Statistics* 30(4):835-847. doi:10.1080/10618600.2021.1904962.

See Also

Fit models with `multiridge()` and extract coefficients with `coef()`.

Examples

```
data <- simulate_data(prob_primary = 1.0)
model <- multiridge(x = data$x_train, y = data$y_train, group = data$group)
beta_hat <- coef(model)
y_hat <- predict(object = model, newx = data$x_test)
```

residuals.cv.corila	<i>Residuals</i>
---------------------	------------------

Description

Residuals

Usage

```
## S3 method for class 'cv.corila'
residuals(object, ...)
```

Arguments

object	object of class "cv.corila"
...	(for compatibility with <code>stats::residuals</code>)

Details

This function extracts the observed and fitted values from the fitted model and calls the internal function `.residuals()` to calculate the residuals.

Value

Returns a numeric vector of length n_0 (one entry for each training observation).

Examples

```
# listing S3 methods
methods(class = "cv.corila")

# simulating data
n <- 10L; p <- 20L; q <- 5L
x <- matrix(rnorm(n * p), nrow = n , ncol = p)
y <- rnorm(n)
group <- rep(seq_len(q), length.out = p)
primary <- as.logical(rbinom(n = p, size = 1L, prob = 0.5))

# fitting the model
object <- cv.corila(x = x, y = y, group = group, primary = primary)

# using S3 methods
coef(object)
predict(object, newx = x)
fitted(object)
residuals(object)
plot(object)
print(object)
summary(object)
```

simulate_data

Data simulation

Description**[Experimental]**

Simulates a predictor matrix, an effect vector and a response vector. The simulated datasets can be used for modelling a response based on grouped and correlated primary and auxiliary predictors.

Usage

```
simulate_data(
  n0 = 50L,
  n1 = 20L,
  p = 30L,
  q = 10L,
  family = "gaussian",
  rho_between = 0,
  rho_within = 0.5,
  probb_primary = 0.5,
  probb_group = 0.5,
  probb_predictor = 0.8,
  signal_strength = 1,
  seed = 1L
)
```

Arguments

n0	number of training observations: positive integer scalar (minimum 1, maximum 10 000)
n1	number of testing observations: non-negative integer scalar (minimum 0, maximum 100 000)
p	number of predictors: positive integer scalar (minimum 1 leads to a single predictor, maximum 1 000)
q	number of predictor groups: positive integer scalar (minimum 1 assigns all predictors to the same group. maximum p assigns each predictor to its own group)
family	character string "gaussian", "binomial", "poisson", or "cox"
rho_between	correlation coefficient for predictors in different groups: positive numeric scalar in the unit interval (minimum 0 leads to uncorrelated predictors between groups, maximum rho_within leads to same correlation between and within groups)
rho_within	correlation coefficient for predictors within the same group: positive numeric scalar in the unit interval (minimum 0 leads to uncorrelated predictors within each group, maximum 1 leads to identical predictors within each group)
prob_primary	probability for each predictor to be primary (rather than auxiliary): numeric scalar in the unit interval (minimum 0 leads to auxiliary predictors only, maximum 1 leads to primary predictors only)
prob_group	probability for each predictor group to be active: numeric scalar in the unit interval (minimum 0 makes all groups inactive, maximum 1 makes all groups active)
prob_predictor	probability for each predictor in an active group to be active: numeric scalar in the unit interval (minimum 0 makes all predictors inactive, maximum 1 makes all predictors in active groups active)
signal_strength	non-negative numeric scalar for multiplying the effect sizes (default: signal_strength=1.0, minimum 0 sets all effect sizes to 0, maximum 2 to avoid undefined values)
seed	random seed for reproducibility: integer scalar (unrestricted)

Details

Use the objects `x_train`, `y_train`, `group`, and `primary` for model training. Estimated coefficients can be compared with `beta`.

Use the object `x_test` for model testing. Predicted values can be compared with `y_test`.

Training and testing observations are named `train_` or `test_`, respectively, followed by a number indexing the observations (e.g., `train_1` or `test_1`).

Primary and auxiliary predictors are named `pri_` or `aux_`, respectively, followed by a number indexing the predictor groups, a point, and a number indexing the predictors within this group (e.g., `pri_1.1` or `aux_1.1`).

Value

Returns a named list with the following slots:

- `x_train`: predictor matrix of the training observations (n_0 rows, p columns)
- `y_train`: response vector of the training observations (length n_0)
- `group`: integer vector indicating the group of the predictors (length p)
- `primary`: logical vector indicating primary (TRUE) and auxiliary (FALSE) predictors (length p)
- `beta`: numeric vector of the effects of the predictors on the response (length p)
- `x_test`: $n_1 \times p$ predictor matrix for the test observations
- `y_test`: response vector for the test observations of length n_1

See Also

This function calls the internal functions `.simulate_predictors()`, `.simulate_effects()`, and `.simulate_response()` for simulating the predictor matrix, the effect vector, or the response vector, respectively.

Examples

```
data <- simulate_data(n0 = 50L, n1 = 20L, p = 30L, q = 10L,  
  family = "gaussian", rho_between = 0.0,  
  rho_within = 0.5, prob_primary = 0.5,  
  signal_strength = 1.0, prob_group = 0.5,  
  prob_predictor = 0.8, seed = 1L)  
utils::str(data, vec.len = 2L)
```

Index

- * **classif**
 - cv.corila, [5](#)
 - multiridge, [11](#)
- * **documentation**
 - corila-package, [2](#)
- * **methods**
 - coef.cv.corila, [3](#)
 - coef.multiridge, [4](#)
 - cv.corila, [5](#)
 - multiridge, [11](#)
 - predict.cv.corila, [15](#)
 - predict.multiridge, [16](#)
- * **models**
 - cv.corila, [5](#)
 - multiridge, [11](#)
- * **regression**
 - cv.corila, [5](#)
 - multiridge, [11](#)
- .backscale(), [3](#), [15](#)
- .combine_slopes(), [3](#)
- .deviance(), [9](#)
- .expand_auxiliary(), [15](#)
- .forescale(), [7](#), [15](#)
- .residuals(), [17](#)
- .set_candidates(), [7](#)
- .simulate_effects(), [20](#)
- .simulate_predictors(), [20](#)
- .simulate_response(), [20](#)
- base::plot, [14](#)
- coef(), [3](#), [8](#), [10](#), [11](#), [13](#), [16](#), [17](#)
- coef.cv.corila, [3](#)
- coef.multiridge, [4](#)
- corila(), [8](#)
- corila-package, [2](#)
- createXXblocks(), [13](#)
- cv.corila, [5](#)
- cv.corila(), [2–4](#), [11](#), [13](#), [16](#)
- CVfolds(), [13](#)
- deviance(), [11](#)
- deviance.cv.corila, [8](#)
- fastCV2(), [13](#)
- fitted(), [10](#), [11](#), [16](#)
- fitted.cv.corila, [9](#)
- glmnet-package, [2](#)
- glmnet::cv.glmnet(), [7](#)
- glmnet::deviance.glmnet(), [9](#)
- glmnet::glmnet(), [7](#)
- here, [8](#)
- IWLSCoxridge(), [12](#), [13](#)
- IWLSridge(), [12](#), [13](#)
- methods, [10](#)
- multiridge, [11](#)
- multiridge(), [2](#), [5](#), [17](#)
- multiridge-package, [2](#), [11](#), [13](#)
- nobs(), [11](#)
- nobs.cv.corila, [13](#)
- optLambdasWrap(), [13](#)
- plot(), [11](#)
- plot.cv.corila, [14](#)
- predict(), [3–5](#), [8](#), [10](#), [11](#), [13](#)
- predict.cv.corila, [15](#)
- predict.multiridge, [16](#)
- print(), [11](#)
- residuals(), [10](#), [11](#)
- residuals.cv.corila, [17](#)
- SigmaFromBlocks(), [13](#)
- simulate_data, [18](#)
- stats::coef, [3](#), [4](#)
- stats::deviance, [8](#)

`stats::fitted`, [10](#)
`stats::nobs`, [13](#)
`stats::predict`, [15](#), [16](#)
`stats::residuals`, [17](#)
`summary()`, [11](#)