Package 'catalytic'

May 8, 2026

Title Tools for Applying Catalytic Priors in Statistical Modeling

Version 0.1.0

Description To improve estimation accuracy and stability in statistical modeling, catalytic prior distributions are employed, integrating observed data with synthetic data generated from a simpler model's predictive distribution. This approach enhances model robustness, stability, and flexibility in complex data scenarios. The catalytic prior distributions are introduced by 'Huang et al.' (2020, <doi:10.1073/pnas.1920913117>), Li and Huang (2023, <doi:10.48550/arXiv.2312.01411>).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Config/testthat/edition 3

Imports coda, invgamma, rlang, rstan, stats, truncnorm, MASS, lme4, quadform, survival, methods

Depends R (>= 3.5.0)

LazyData true

Suggests knitr, rmarkdown, testthat, pROC

VignetteBuilder knitr

NeedsCompilation no

Author Yitong Wu [aut] (ORCID: <https://orcid.org/0009-0000-8683-5129>), Dongming Huang [aut, cre] (ORCID: <https://orcid.org/0000-0003-4265-7708>), Weihao Li [aut], Ministry of Education, Singapore [fnd] (The development of this package is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 A-8000466-00-00 (FY2022).)

Maintainer Dongming Huang <huang.dongming@nus.edu.sg>

Repository CRAN

Date/Publication 2024-12-18 16:10:02 UTC

Contents

`cat_cox` *Catalytic Cox Proportional Hazards Model (COX) Fitting Function with Fixed Tau*

Description

Fits a Catalytic Cox proportional hazards model for survival data with specified variance parameters and iterative coefficient estimation, with either `CRE` (Catalytic-regularized Estimator) or `WME` (Weighted Mixture Estimator) methods.

Usage

```
cat_cox(
  formula,
  cat_init,
  tau = NULL,
  method = c("CRE", "WME"),
  init_coefficients = NULL,
  tol = 1e-05,
  max_iter = 25
)
```

Arguments

formula	A formula specifying the Cox model. Should at least include response variables (e.g. ~ .).
cat_init	A list generated from cat_cox_initialization.
tau	Optional numeric scalar controlling the weight of the synthetic data in the coefficient estimation, defaults to the number of predictors.
method	The estimation method, either "CRE" (Catalytic-regularized Estimator) or "WME" (Weighted Mixture Estimator).
init_coefficients	Initial coefficient values before iteration. Defaults to zero if not provided (if using CRE).
tol	Convergence tolerance for iterative methods. Default is 1e-5 (if using CRE).
max_iter	Maximum number of iterations allowed for convergence. Default is 25 (if using CRE).

Value

A list containing the values of all the arguments and the following components:

coefficients	Estimated coefficient vector.
model	Fitted Cox model object (if using WME).
iteration_log	Matrix logging variance and coefficient values for each iteration(if using CRE).
iter	Number of iterations (if using CRE).

Examples

```
library(survival)
data("cancer")
cancer$status[cancer$status == 1] <- 0
cancer$status[cancer$status == 2] <- 1

cat_init <- cat_cox_initialization(
  formula = Surv(time, status) ~ 1, # formula for simple model
  data = cancer,
  syn_size = 100, # Synthetic data size
```

```
  hazard_constant = 0.1, # Hazard rate value
  entry_points = rep(0, nrow(cancer)), # Entry points of each observation
  x_degree = rep(1, ncol(cancer) - 2), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model_cre <- cat_cox(
  formula = ~.,
  cat_init = cat_init, # Only accept object generated from `cat_cox_initialization`
  tau = 1, # Weight for synthetic data
  method = "CRE", # Choose from `"CRE"` or `"WME"`
 init_coefficients = rep(0, ncol(cat_init$x)), # Initial coefficient values (Only for `CRE`)
  tol = 1e-1, # Tolerance for convergence criterion  (Only for `CRE`)
  max_iter = 3 # Maximum number of iterations for convergence  (Only for `CRE`)
)
cat_model_cre

cat_model_wme <- cat_cox(
  formula = ~.,
  cat_init = cat_init, # Only accept object generated from `cat_cox_initialization`
  tau = 1, # Weight for synthetic data
  method = "WME"
)
cat_model_wme
```

`cat_cox_bayes` *Bayesian Estimation for Catalytic Cox Proportional-Hazards Model (COX) with Fixed tau*

Description

This function performs Bayesian estimation for a catalytic Cox proportional-hazards model (COX) using RStan by given a single tau value. It allows users to estimate the coefficients and cumulative baseline hazard increments over specified time intervals with Bayesian sampling.

Usage

```
cat_cox_bayes(
  formula,
  cat_init,
  tau = NULL,
  hazard_beta = 2,
  chains = 4,
  iter = 2000,
  warmup = 1000,
  ...
)
```

Arguments

formula	A formula specifying the Cox model. Should at least include response variables (e.g. ~.).
cat_init	A list generated from cat_cox_initialization.
tau	Optional numeric scalar controlling the weight of the synthetic data in the coefficient estimation, defaults to ncol(cat_init$obs_x).
hazard_beta	Numeric, default 2. Shape parameter for the Gamma distribution in the hazard model.
chains	Integer, default 4. Number of Markov chains to be run in the RStan sampling.
iter	Integer, default 2000. Number of iterations per chain in the RStan sampling.
warmup	Integer, default 1000. Number of warm-up (or burn-in) iterations for each chain.
...	Additional arguments passed to RStan's rstan::sampling function.

Value

A list containing the values of all the arguments and the following components:

stan_data	A data list used for fitting RStan sampling model.
stan_model	Compiled RStan model object for Cox regression.
stan_sample_model	Fitted RStan sampling model containing posterior samples.
coefficients	Mean posterior estimates of model coefficients from stan_sample_model.
increment_cumulative_baseline_hazard	Mean posterior estimates of Estimated cumulative hazard increments over time intervals from stan_sample_model.

Examples

```
library(survival)
data("cancer")
cancer$status[cancer$status == 1] <- 0
cancer$status[cancer$status == 2] <- 1

cat_init <- cat_cox_initialization(
  formula = Surv(time, status) ~ 1, # formula for simple model
  data = cancer,
  syn_size = 100, # Synthetic data size
  hazard_constant = 0.1, # Hazard rate value
  entry_points = rep(0, nrow(cancer)), # Entry points of each observation
  x_degree = rep(1, ncol(cancer) - 2), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_cox_bayes(
  formula = ~., # Should at least include response variables
  cat_init = cat_init, # Only accept object generated from `cat_cox_initialization`
```

```
  tau = 1, # Weight for synthetic data
  hazard_beta = 2, # Shape parameter for the Gamma distribution in the hazard model
  chains = 1, # Number of Markov chains to be run in the RStan sampling
  iter = 10, # Number of iterations per chain in the RStan sampling
  warmup = 5 # Number of warm-up (or burn-in) iterations for each chain
)
cat_model
```

`cat_cox_bayes_joint` *Bayesian Estimation for Catalytic Cox Proportional-Hazards Model (COX) with adaptive tau*

Description

This function performs Bayesian estimation for a catalytic Cox proportional-hazards model (COX) using RStan by using adaptive tau. It allows users to estimate the coefficients and cumulative baseline hazard increments over specified time intervals with Bayesian sampling.

Usage

```
cat_cox_bayes_joint(
  formula,
  cat_init,
  hazard_beta = 2,
  tau_alpha = 2,
  tau_gamma = 1,
  chains = 4,
  iter = 2000,
  warmup = 1000,
  ...
)
```

Arguments

`formula`	A formula specifying the Cox model. Should at least include response variables (e.g. `~ .`).
`cat_init`	A list generated from `cat_cox_initialization`.
`hazard_beta`	Numeric, default `2`. Shape parameter for the Gamma distribution in the hazard model.
`tau_alpha`	Numeric, defaults `2`. Scalar for the shape parameter of the Gamma-like function for tau.
`tau_gamma`	Numeric, defaults `1`. Scalar for the scale parameter of the Gamma-like function for tau.
`chains`	Integer, default `4`. Number of Markov chains to be run in the RStan sampling.
`iter`	Integer, default `2000`. Number of iterations per chain in the RStan sampling.
`warmup`	Integer, default `1000`. Number of warm-up (or burn-in) iterations for each chain.
`...`	Additional arguments passed to RStan's `rstan::sampling` function.

Value

A list containing the values of all the arguments and the following components:

`stan_data` A data list used for fitting RStan sampling model.

`stan_model` Compiled RStan model object for Cox regression.

`stan_sample_model`
Fitted RStan sampling model containing posterior samples.

`tau` Mean posterior estimates of tau value from `stan_sample_model`.

`coefficients` Mean posterior estimates of model coefficients from `stan_sample_model`.

`increment_cumulative_baseline_hazard`
Mean posterior estimates of Estimated cumulative hazard increments over time intervals from `stan_sample_model`.

Examples

```
library(survival)
data("cancer")
cancer$status[cancer$status == 1] <- 0
cancer$status[cancer$status == 2] <- 1

cat_init <- cat_cox_initialization(
  formula = Surv(time, status) ~ 1, # formula for simple model
  data = cancer,
  syn_size = 100, # Synthetic data size
  hazard_constant = 0.1, # Hazard rate value
  entry_points = rep(0, nrow(cancer)), # Entry points of each observation
  x_degree = rep(1, ncol(cancer) - 2), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_cox_bayes_joint(
  formula = ~., # Should at least include response variables
  cat_init = cat_init, # Only accept object generated from `cat_cox_initialization`
  hazard_beta = 2, # Shape parameter for the Gamma distribution in the hazard model
  tau_alpha = 2, # Shape parameter of the Gamma-like function for tau
  tau_gamma = 1, # Scale parameter of the Gamma-like function for tau
  chains = 1, # Number of Markov chains to be run in the RStan sampling
  iter = 10, # Number of iterations per chain in the RStan sampling
  warmup = 5 # Number of warm-up (or burn-in) iterations for each chain
)
cat_model
```

`cat_cox_initialization` *Initialization for Catalytic Cox proportional hazards model (COX)*

Description

This function prepares and initializes a catalytic Cox proportional hazards model by processing input data, extracting necessary variables, generating synthetic datasets, and fitting a model.

Usage

```
cat_cox_initialization(
  formula,
  data,
  syn_size = NULL,
  hazard_constant = NULL,
  entry_points = NULL,
  x_degree = NULL,
  resample_only = FALSE,
  na_replace = stats::na.omit
)
```

Arguments

`formula`	A formula specifying the Cox model. Should include response and predictor variables.
`data`	A data frame containing the data for modeling.
`syn_size`	An integer specifying the size of the synthetic dataset to be generated. Default is four times the number of predictor columns.
`hazard_constant`	A constant hazard rate for generating synthetic time data if not using a fitted Cox model. Default is NULL and will calculate in function.
`entry_points`	A numeric vector for entry points of each observation. Default is NULL.
`x_degree`	A numeric vector indicating the degree for polynomial expansion of predictors. Default is 1 for each predictor.
`resample_only`	A logical indicating whether to perform resampling only. Default is FALSE.
`na_replace`	A function to handle NA values in the data. Default is `stats::na.omit`.

Value

A list containing the values of all the input arguments and the following components:

- **Function Information**:
  - `function_name`: The name of the function, "cat_cox_initialization".
  - `time_col_name`: The name of the time variable in the dataset.

  - `status_col_name`: The name of the status variable (event indicator) in the dataset.
  - `simple_model`: If the formula has no predictors, a constant hazard rate model is used; otherwise, a fitted Cox model object.
- **Observation Data Information**:
  - `obs_size`: Number of observations in the original dataset.
  - `obs_data`: Data frame of standardized observation data.
  - `obs_x`: Predictor variables for observed data.
  - `obs_time`: Observed survival times.
  - `obs_status`: Event indicator for observed data.
- **Synthetic Data Information**:
  - `syn_size`: Number of synthetic observations generated.
  - `syn_data`: Data frame of synthetic predictor and response variables.
  - `syn_x`: Synthetic predictor variables.
  - `syn_time`: Synthetic survival times.
  - `syn_status`: Event indicator for synthetic data (defaults to 1).
  - `syn_x_resample_inform`: Information about resampling methods for synthetic predictors:
    * Coordinate: Preserves the original data values as reference coordinates during processing.
    * Deskewing: Adjusts the data distribution to reduce skewness and enhance symmetry.
    * Smoothing: Reduces noise in the data to stabilize the dataset and prevent overfitting.
    * Flattening: Creates a more uniform distribution by modifying low-frequency categories in categorical variables.
    * Symmetrizing: Balances the data around its mean to improve statistical properties for model fitting.
- **Whole Data Information**:
  - `size`: Total number of combined original and synthetic observations.
  - `data`: Data frame combining original and synthetic datasets.
  - `x`: Combined predictor variables from original and synthetic data.
  - `time`: Combined survival times from original and synthetic data.
  - `status`: Combined event indicators from original and synthetic data.

Examples

```
library(survival)
data("cancer")
cancer$status[cancer$status == 1] <- 0
cancer$status[cancer$status == 2] <- 1

cat_init <- cat_cox_initialization(
  formula = Surv(time, status) ~ 1, # formula for simple model
  data = cancer,
  syn_size = 100, # Synthetic data size
  hazard_constant = NULL, # Hazard rate value
  entry_points = rep(0, nrow(cancer)), # Entry points of each observation
```

```
  x_degree = rep(1, ncol(cancer) - 2), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)
cat_init
```

`cat_cox_tune` *Catalytic Cox Proportional-Hazards Model (COX) Fitting Function by Tuning tau from a Sequence of tau Values*

Description

This function tunes a catalytic Cox proportional-hazards model (COX) by performing cross-validation to estimate the optimal value of the tuning parameter `tau`. It finally uses the optimal tau value in the `cat_cox` function for model fitting.

Usage

```
cat_cox_tune(
  formula,
  cat_init,
  method = c("CRE", "WME"),
  tau_seq = NULL,
  cross_validation_fold_num = 5,
  ...
)
```

Arguments

`formula`	A formula specifying the Cox model. Should at least include response variables (e.g. `~.`).
`cat_init`	A list generated from `cat_cox_initialization`.
`method`	The estimation method, either `"CRE"` (Catalytic-regularized Estimator) or `"WME"` (Weighted Mixture Estimator).
`tau_seq`	A numeric vector specifying the sequence of `tau` values to be tested. If NULL, a default sequence is generated based on the number of predictors.
`cross_validation_fold_num`	An integer representing the number of folds for cross-validation. Defaults to 5.
`...`	Additional arguments passed to the `cat_cox` function for model fitting.

Value

A list containing the values of all the arguments and the following components:

`tau`	he optimal `tau` value determined from cross-validation.
`model`	The fitted lmer model object by using the optimal `tau` value.

`coefficients` Coefficients of the fitted model by using the optimal `tau` value.

`likelihood_list`
Average likelihood value for each `tau` value.

Examples

```
library(survival)
data("cancer")
cancer$status[cancer$status == 1] <- 0
cancer$status[cancer$status == 2] <- 1

cat_init <- cat_cox_initialization(
  formula = Surv(time, status) ~ 1, # formula for simple model
  data = cancer,
  syn_size = 100, # Synthetic data size
  hazard_constant = 0.1, # Hazard rate value
  entry_points = rep(0, nrow(cancer)), # Entry points of each observation
  x_degree = rep(1, ncol(cancer) - 2), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_cox_tune(
  formula = ~., # Should at least include response variables
  cat_init = cat_init, # Only accept object generated from `cat_cox_initialization`
  tau_seq = c(1, 2), # Vector of weights for synthetic data
  cross_validation_fold_num = 5 # number of folds for cross-validation
)
cat_model
```

---

cat_glm *Catalytic Generalized Linear Models (GLMs) Fitting Function with Fixed Tau*

Description

Fits a Catalytic Generalized Linear Models (GLMs) by using observed and synthetic data.

Usage

```
cat_glm(formula, cat_init, tau = NULL)
```

Arguments

`formula`	A formula specifying the GLMs. Should at least include response variables (e.g. ~ .).
`cat_init`	A list generated from `cat_glm_initialization`.
`tau`	Optional numeric scalar controlling the weight of the synthetic data in the coefficient estimation. Defaults to the number of predictors / 4 for Gaussian models or the number of predictors otherwise.

Value

A list containing the values of all the arguments and the following components:

`coefficients`	Estimated coefficient vector.
`model`	Fitted GLMs object (`stats::glm`).

Examples

```
gaussian_data <- data.frame(
  X1 = stats::rnorm(10),
  X2 = stats::rnorm(10),
  Y = stats::rnorm(10)
)

cat_init <- cat_glm_initialization(
  formula = Y ~ 1, # formula for simple model
  data = gaussian_data,
  syn_size = 100, # Synthetic data size
  custom_variance = NULL, # User customized variance value
  gaussian_known_variance = TRUE, # Indicating whether the data variance is known
  x_degree = c(1, 1), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_glm(
  formula = ~.,
  cat_init = cat_init, # Only accept object generated from `cat_glm_initialization`
  tau = 1 # Weight for synthetic data
)
cat_model
```

---

cat_glm_bayes *Bayesian Estimation for Catalytic Generalized Linear Models (GLMs) with Fixed tau*

Description

Fits a Bayesian generalized linear model using synthetic and observed data based on an initial data structure, formula, and other model specifications. Supports only Gaussian and Binomial distributions in the GLM family.

Usage

```
cat_glm_bayes(
  formula,
  cat_init,
  tau = NULL,
  chains = 4,
```

```
  iter = 2000,
  warmup = 1000,
  algorithm = "NUTS",
  gaussian_variance_alpha = NULL,
  gaussian_variance_beta = NULL,
  ...
)
```

Arguments

formula	A formula specifying the GLMs. Should at least include response variables (e.g. ~.).
cat_init	A list generated from cat_glm_initialization.
tau	Optional numeric scalar controlling the weight of the synthetic data in the coefficient estimation. Defaults to the number of predictors / 4 for Gaussian models or the number of predictors otherwise.
chains	Number of Markov chains to run. Default is 4.
iter	Total number of iterations per chain. Default is 2000.
warmup	Number of warm-up iterations per chain (discarded from final analysis). Default is 1000.
algorithm	The sampling algorithm to use in rstan::sampling. Default is "NUTS" (No-U-Turn Sampler).
gaussian_variance_alpha	The shape parameter for the inverse-gamma prior on variance if the variance is unknown in Gaussian models. Defaults to the number of predictors.
gaussian_variance_beta	The scale parameter for the inverse-gamma prior on variance if the variance is unknown in Gaussian models. Defaults to the number of predictors times variance of observation response.
...	Additional parameters to pass to rstan::sampling.

Value

A list containing the values of all the arguments and the following components:

stan_data	The data list used for fitting RStan sampling model.
stan_model	Compiled RStan model object for GLMs.
stan_sample_model	Fitted RStan sampling model containing posterior samples.
coefficients	Mean posterior estimates of model coefficients from stan_sample_model.

Examples

```
gaussian_data <- data.frame(
  X1 = stats::rnorm(10),
  X2 = stats::rnorm(10),
```

```
  Y = stats::rnorm(10)
)

cat_init <- cat_glm_initialization(
  formula = Y ~ 1, # formula for simple model
  data = gaussian_data,
  syn_size = 100, # Synthetic data size
  custom_variance = NULL, # User customized variance value
  gaussian_known_variance = FALSE, # Indicating whether the data variance is unknown
  x_degree = c(1, 1), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_glm_bayes(
  formula = ~.,
  cat_init = cat_init, # Only accept object generated from `cat_glm_initialization`
  tau = 1, # Weight for synthetic data
  chains = 1, # Number of Markov chains to be run in the RStan sampling
  iter = 10, # Number of iterations per chain in the RStan sampling
  warmup = 5, # Number of warm-up (or burn-in) iterations for each chain
  algorithm = "NUTS", # Sampling algorithm to use in \code{rstan::sampling}
 gaussian_variance_alpha = 1, # The shape parameter for the inverse-gamma prior for variance
  gaussian_variance_beta = 2 # The scale parameter for the inverse-gamma prior for variance
)
cat_model
```

`cat_glm_bayes_joint` *Bayesian Estimation for Catalytic Generalized Linear Models (GLMs) with adaptive tau*

Description

This function performs Bayesian estimation for a catalytic Generalized Linear Models (GLMs) using RStan by using adaptive tau. It supports both Gaussian and Binomial family models, enabling flexibility in prior specifications and algorithm configurations.

Usage

```
cat_glm_bayes_joint(
  formula,
  cat_init,
  chains = 4,
  iter = 2000,
  warmup = 1000,
  algorithm = "NUTS",
  tau_alpha = 2,
  tau_gamma = 1,
```

```
  binomial_tau_lower = 0.05,
  binomial_joint_theta = FALSE,
  binomial_joint_alpha = FALSE,
  gaussian_variance_alpha = NULL,
  gaussian_variance_beta = NULL,
  ...
)
```

Arguments

formula	A formula specifying the GLMs. Should at least include response variables (e.g. ~ .).
cat_init	A list generated from cat_glm_initialization.
chains	Number of Markov chains to run. Default is 4.
iter	Total number of iterations per chain. Default is 2000.
warmup	Number of warm-up iterations per chain (discarded from final analysis). Default is 1000.
algorithm	The sampling algorithm to use in rstan::sampling. Default is "NUTS" (No-U-Turn Sampler).
tau_alpha	Shape parameter of the prior for tau. Default is 2.
tau_gamma	Scale parameter of the prior for tau. Default is 1.
binomial_tau_lower	A numeric lower bound for tau in Binomial models. Default is 0.05.
binomial_joint_theta	Logical; if TRUE, uses joint theta (theta = 1/tau) in Binomial models. Default is FALSE.
binomial_joint_alpha	Logical; if TRUE, uses joint alpha (adaptive tau_alpha) in Binomial models. Default is FALSE. To activate this feature, both binomial_joint_theta = TRUE and binomial_joint_alpha = TRUE must be set.
gaussian_variance_alpha	The shape parameter for the inverse-gamma prior on variance if the variance is unknown in Gaussian models. Defaults to the number of predictors.
gaussian_variance_beta	The scale parameter for the inverse-gamma prior on variance if the variance is unknown in Gaussian models. Defaults to the number of predictors times variance of observation response.
...	Additional parameters to pass to rstan::sampling.

Value

A list containing the values of all the arguments and the following components:

stan_data	A data list used for fitting RStan sampling model.
stan_model	Compiled RStan model object for GLMs.

`stan_sample_model`
: Fitted RStan sampling model containing posterior samples.

`coefficients`
: Mean posterior estimates of model coefficients from `stan_sample_model`.

`tau`
: Mean posterior of tau (or transformed theta if applicable).

Examples

```
gaussian_data <- data.frame(
  X1 = stats::rnorm(10),
  X2 = stats::rnorm(10),
  Y = stats::rnorm(10)
)

cat_init <- cat_glm_initialization(
  formula = Y ~ 1, # formula for simple model
  data = gaussian_data,
  syn_size = 100, # Synthetic data size
  custom_variance = NULL, # User customized variance value
  gaussian_known_variance = FALSE, # Indicating whether the data variance is unknown
  x_degree = c(1, 1), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_glm_bayes_joint(
  formula = ~.,
  cat_init = cat_init, # Only accept object generated from `cat_glm_initialization`
  chains = 1, # Number of Markov chains to be run in the RStan sampling
  iter = 10, # Number of iterations per chain in the RStan sampling
  warmup = 5, # Number of warm-up (or burn-in) iterations for each chain
  algorithm = "NUTS", # Sampling algorithm to use in \code{rstan::sampling}
  tau_alpha = 1, # Shape parameter of the prior for tau
  tau_gamma = 2, # Scale parameter of the prior for tau
  binomial_tau_lower = 0.05, # Lower bound for tau in Binomial models.
  binomial_joint_theta = FALSE, # Indicator for using joint theta for Binomial models
  binomial_joint_alpha = FALSE, # Indicator for using oint alpha for Binomial models
 gaussian_variance_alpha = 1, # The shape parameter for the inverse-gamma prior for variance
 gaussian_variance_beta = 2 # The scale parameter for the inverse-gamma prior for variance
)
cat_model
```

---

cat_glm_bayes_joint_gibbs

Bayesian Estimation with Gibbs Sampling for Catalytic Generalized Linear Models (GLMs) Binomial Family for Coefficients and tau

---

Description

This function uses Gibbs sampling to estimate a Bayesian GLMs Binomial Family, where both the coefficients and tau parameter are jointly sampled. tau is updated via a gamma distribution, while coefficients are updated using Hamiltonian Monte Carlo (HMC) sampling. The model allows for progress updates, warm-up iterations, and initial coefficient estimation based on initial tau value.

Usage

```
cat_glm_bayes_joint_gibbs(
  formula,
  cat_init,
  iter = 1000,
  warmup = 500,
  coefs_iter = 5,
  tau_0 = NULL,
  tau_alpha = 2,
  tau_gamma = 1,
  refresh = TRUE
)
```

Arguments

formula	A formula specifying the GLMs. Should at least include response variables (e.g. ~ .).
cat_init	A list generated from cat_glm_initialization.
iter	Integer; the number of Gibbs sampling iterations (default = 1000).
warmup	Integer; the number of initial iterations for warm-up (default = 500).
coefs_iter	Integer; the number of iterations for the HMC step to update coefficients.
tau_0	Initial value for tau; defaults to the number of predictors / 4 if NULL.
tau_alpha	Shape parameter for the gamma distribution when updating tau. Default is 2.
tau_gamma	Scale parameter for the gamma distribution when updating tau. Default is 1.
refresh	Logical; if TRUE, displays sampling progress. Default is TRUE.

Value

A list containing the values of all the arguments and the following components:

gibbs_iteration_log	Matrix containing the coefficients and tau values from each Gibbs iteration.
inform_df	Summary statistics of each parameter, including mean, standard error, quantiles, and effective sample size.
tau	Mean of sampled tau values.
coefficients	Mean of sampled coefficient values.

Examples

```
binomial_data <- data.frame(
  X1 = stats::rnorm(10),
  X2 = stats::rnorm(10),
  Y = stats::rbinom(10, 1, 0.5)
)

cat_init <- cat_glm_initialization(
  formula = Y ~ 1, # formula for simple model
  data = binomial_data,
  family = binomial,
  syn_size = 100, # Synthetic data size
  custom_variance = NULL, # User customized variance value
  gaussian_known_variance = FALSE, # Indicating whether the data variance is unknown
  x_degree = c(1, 1), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_glm_bayes_joint_gibbs(
  formula = ~.,
  cat_init = cat_init, # Only accept object generated from `cat_glm_initialization`
  iter = 10, # Number of Gibbs sampling iterations
  warmup = 5, # Number of warm-up (or burn-in) iterations for initial iterations
  coefs_iter = 2, # Number of iterations for the HMC step to update coefficients
  tau_alpha = 1, # Shape parameter for the gamma distribution when updating tau
  tau_gamma = 2, # Scale parameter for the gamma distribution when updating tau
  refresh = TRUE # Indicator for displaying sampling progress
)
cat_model
```

`cat_glm_initialization` *Initialization for Catalytic Generalized Linear Models (GLMs)*

Description

This function prepares and initializes a catalytic Generalized Linear Models (GLMs) by processing input data, extracting necessary variables, generating synthetic datasets, and fitting a model.

Usage

```
cat_glm_initialization(
  formula,
  family = "gaussian",
  data,
  syn_size = NULL,
  custom_variance = NULL,
```

```
  gaussian_known_variance = FALSE,
  x_degree = NULL,
  resample_only = FALSE,
  na_replace = stats::na.omit
)
```

Arguments

formula	A formula specifying the GLMs. Should include response and predictor variables.
family	The type of GLM family. Defaults to Gaussian.
data	A data frame containing the data for modeling.
syn_size	An integer specifying the size of the synthetic dataset to be generated. Default is four times the number of predictor columns.
custom_variance	A custom variance value to be applied if using a Gaussian model. Defaults to NULL.
gaussian_known_variance	A logical value indicating whether the data variance is known. Defaults to FALSE. Only applicable to Gaussian family.
x_degree	A numeric vector indicating the degree for polynomial expansion of predictors. Default is 1 for each predictor.
resample_only	A logical indicating whether to perform resampling only. Default is FALSE.
na_replace	A function to handle NA values in the data. Default is stats::na.omit.

Value

A list containing the values of all the input arguments and the following components:

- **Function Information**
  - function_name: The name of the function, "cat_glm_initialization".
  - y_col_name: The name of the response variable in the dataset.
  - simple_model: An object of class stats::glm, representing the fitted model for generating synthetic response from the original data.
- **Observation Data Information**
  - obs_size: Number of observations in the original dataset.
  - obs_data: Data frame of standardized observation data.
  - obs_x: Predictor variables for observed data.
  - obs_y: Response variable for observed data.
- **Synthetic Data Information**
  - syn_size: Number of synthetic observations generated.
  - syn_data: Data frame of synthetic predictor and response variables.
  - syn_x: Synthetic predictor variables.
  - syn_y: Synthetic response variable.

  - syn_x_resample_inform: Information about resampling methods for synthetic predictors:
    * Coordinate: Preserves the original data values as reference coordinates during processing.
    * Deskewing: Adjusts the data distribution to reduce skewness and enhance symmetry.
    * Smoothing: Reduces noise in the data to stabilize the dataset and prevent overfitting.
    * Flattening: Creates a more uniform distribution by modifying low-frequency categories in categorical variables.
    * Symmetrizing: Balances the data around its mean to improve statistical properties for model fitting.

- **Whole Data Information**
  - size: Total number of combined original and synthetic observations.
  - data: Data frame combining original and synthetic datasets.
  - x: Combined predictor variables from original and synthetic data.
  - y: Combined response variable from original and synthetic data.

Examples

```
gaussian_data <- data.frame(
  X1 = stats::rnorm(10),
  X2 = stats::rnorm(10),
  Y = stats::rnorm(10)
)

cat_init <- cat_glm_initialization(
  formula = Y ~ 1, # formula for simple model
  data = gaussian_data,
  syn_size = 100, # Synthetic data size
  custom_variance = NULL, # User customized variance value
  gaussian_known_variance = TRUE, # Indicating whether the data variance is known
  x_degree = c(1, 1), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)
cat_init
```

---

cat_glm_tune *Catalytic Generalized Linear Models (GLMs) Fitting Function by Tuning tau from a Sequence of tau Values*

Description

This function tunes a catalytic catalytic Generalized Linear Models (GLMs) by performing specified risk estimate method to estimate the optimal value of the tuning parameter tau. The resulting cat_glm_tune object encapsulates the fitted model, including estimated coefficients and family information, facilitating further analysis.

Usage

```
cat_glm_tune(
  formula,
  cat_init,
  risk_estimate_method = c("parametric_bootstrap", "cross_validation",
    "mallowian_estimate", "steinian_estimate"),
  discrepancy_method = c("mean_square_error", "mean_classification_error",
    "logistic_deviance"),
  tau_seq = NULL,
  tau_0 = NULL,
  parametric_bootstrap_iteration_times = 100,
  cross_validation_fold_num = 5
)
```

Arguments

formula	A formula specifying the GLMs. Should at least include response variables (e.g. ~ .).
cat_init	A list generated from cat_glm_initialization.
risk_estimate_method	Method for risk estimation, chosen from "parametric_bootstrap", "cross_validation", "mallows_estimate", "steinian_estimate". Depends on the size of the data if not provided.
discrepancy_method	Method for discrepancy calculation, chosen from "mean_square_error", "mean_classification_error", "logistic_deviance". Depends on the family if not provided.
tau_seq	Vector of numeric values for down-weighting synthetic data. Defaults to a sequence around one fourth of the number of predictors for gaussian and the number of predictors for binomial.
tau_0	Initial tau value used for discrepancy calculation in risk estimation. Defaults to one fourth of the number of predictors for binomial and 1 for gaussian.
parametric_bootstrap_iteration_times	Number of bootstrap iterations for "parametric_bootstrap" risk estimation. Defaults to 100.
cross_validation_fold_num	Number of folds for "cross_validation" risk estimation.. Defaults to 5.

Value

A list containing the values of all the arguments and the following components:

tau	Optimal tau value determined through tuning.
model	Fitted GLM model object with the optimal tau value.
coefficients	Estimated coefficients from the model fitted by the optimal tau value.
risk_estimate_list	Collected risk estimates for each tau.

Examples

```
gaussian_data <- data.frame(
  X1 = stats::rnorm(10),
  X2 = stats::rnorm(10),
  Y = stats::rnorm(10)
)

cat_init <- cat_glm_initialization(
  formula = Y ~ 1, # formula for simple model
  data = gaussian_data,
  syn_size = 100, # Synthetic data size
  custom_variance = NULL, # User customized variance value
  gaussian_known_variance = TRUE, # Indicating whether the data variance is known
  x_degree = c(1, 1), # Degrees for polynomial expansion of predictors
  resample_only = FALSE, # Whether to perform resampling only
  na_replace = stats::na.omit # How to handle NA values in data
)

cat_model <- cat_glm_tune(
  formula = ~.,
  cat_init = cat_init, # Only accept object generated from `cat_glm_initialization`
  risk_estimate_method = "parametric_bootstrap",
  discrepancy_method = "mean_square_error",
  tau_seq = c(1, 2), # Weight for synthetic data
  tau_0 = 2,
  parametric_bootstrap_iteration_times = 20, # Number of bootstrap iterations
  cross_validation_fold_num = 5 # Number of folds
)
cat_model
```

---

cat_lmm *Catalytic Linear Mixed Model (LMM) Fitting Function with fixed tau*

Description

Fits a Catalytic linear mixed model (LMM) for observation and synthetic data with specified variance parameters and iterative coefficient estimation. This function initializes model parameters, sorts synthetic data, calculates Eigen-decomposition, and iterative optimizes variance and coefficient values to convergence, by a single given tau value. (Only consider one random effect variance)

Usage

```
cat_lmm(
  cat_init,
  tau = NULL,
  residual_variance_0 = 1,
  random_effect_variance_0 = 1,
  coefs_0 = NULL,
  optimize_domain = c(0, 30),
```

```
  max_iter = 500,
  tol = 1e-08
)
```

Arguments

cat_init	A list generated from cat_lmm_initialization.
tau	Optional numeric scalar controlling the weight of the synthetic data in the coefficient estimation, defaults to ncol(cat_init$obs_x) / 4.
residual_variance_0	Initial value for residual variance, default is 1.
random_effect_variance_0	Initial value for random effect variance, default is 1.
coefs_0	Optional initial coefficient vector, default is NULL which initializes randomly.
optimize_domain	Numeric vector of length 2 defining optimization range for variance parameters, default is c(0, 30).
max_iter	Integer specifying maximum number of iterations for convergence, default is 500.
tol	Tolerance for convergence criterion, default is 1e-08.

Value

A list containing the values of all the arguments and the following components:

coefficients	Estimated coefficient vector.
iteration_log	Matrix logging variance and coefficient values for each iteration.

Examples

```
data(mtcars)
cat_init <- cat_lmm_initialization(
  formula = mpg ~ wt + (1 | cyl), # formula for simple model
  data = mtcars,
  x_cols = c("wt"), # Fixed effects
  y_col = "mpg", # Response variable
  z_cols = c("disp", "hp", "drat", "qsec", "vs", "am", "gear", "carb"), # Random effects
  group_col = "cyl", # Grouping column
  syn_size = 100, # Synthetic data size
  resample_by_group = FALSE, # Resampling option
  resample_only = FALSE, # Resampling method
  na_replace = mean # NA replacement method
)

cat_model <- cat_lmm(
  cat_init = cat_init, # Only accept object generated from cat_lmm_initialization
  tau = 1, # Weight for synthetic data
  residual_variance_0 = 1, # Initial value for residual variance
  random_effect_variance_0 = 1, # Initial value for random effect variance
```

```
  coefs_0 = c(1), # Initial coefficient vector
  optimize_domain = c(0, 10), # Optimization range for residual and random effect variance
  max_iter = 2, # Maximum number of iterations for convergence
  tol = 1e-01 # Tolerance for convergence criterion
)
cat_model
```

`cat_lmm_initialization` *Initialization for Catalytic Linear Mixed Model (LMM)*

Description

This function prepares and initializes a catalytic linear mixed model by processing input data, extracting necessary variables, generating synthetic datasets, and fitting a model. (Only consider one random effect variance)

Usage

```
cat_lmm_initialization(
  formula,
  data,
  x_cols,
  y_col,
  z_cols,
  group_col = NULL,
  syn_size = NULL,
  resample_by_group = FALSE,
  resample_only = FALSE,
  na_replace = mean
)
```

Arguments

`formula`	A formula specifying the model. Should include response and predictor variables.
`data`	A data frame containing the data for modeling.
`x_cols`	A character vector of column names for fixed effects (predictors).
`y_col`	A character string for the name of the response variable.
`z_cols`	A character vector of column names for random effects.
`group_col`	A character string for the grouping variable (optional). If not given (NULL), it is extracted from the formula.
`syn_size`	An integer specifying the size of the synthetic dataset to be generated, default is length(x_cols) * 4.
`resample_by_group`	A logical indicating whether to resample by group, default is FALSE.

`resample_only` A logical indicating whether to perform resampling only, default is FALSE.

`na_replace` A function to replace NA values in the data, default is mean.

Value

A list containing the values of all the input arguments and the following components:

- **Function Information**:
  - `function_name`: A character string representing the name of the function, "cat_lmm_initialization".
  - `simple_model`: An object of class `lme4::lmer` or `stats::lm`, representing the fitted model for generating synthetic response from the original data.
- **Observation Data Information**:
  - `obs_size`: An integer representing the number of observations in the original dataset.
  - `obs_data`: The original data used for fitting the model, returned as a data frame.
  - `obs_x`: A data frame containing the standardized predictor variables from the original dataset.
  - `obs_y`: A numeric vector of the standardized response variable from the original dataset.
  - `obs_z`: A data frame containing the standardized random effect variables from the original dataset.
  - `obs_group`: A numeric vector representing the grouping variable for the original observations.
- **Synthetic Data Information**:
  - `syn_size`: An integer representing the number of synthetic observations generated.
  - `syn_data`: A data frame containing the synthetic dataset, combining synthetic predictor and response variables.
  - `syn_x`: A data frame containing the synthetic predictor variables.
  - `syn_y`: A numeric vector of the synthetic response variable values.
  - `syn_z`: A data frame containing the synthetic random effect variables.
  - `syn_group`: A numeric vector representing the grouping variable for the synthetic observations.
  - `syn_x_resample_inform`: A data frame containing information about the resampling process for synthetic predictors:
    - Coordinate: Preserves the original data values as reference coordinates during processing.
    - Deskewing: Adjusts the data distribution to reduce skewness and enhance symmetry.
    - Smoothing: Reduces noise in the data to stabilize the dataset and prevent overfitting.
    - Flattening: Creates a more uniform distribution by modifying low-frequency categories in categorical variables.
    - Symmetrizing: Balances the data around its mean to improve statistical properties for model fitting.
  - `syn_z_resample_inform`: A data frame containing information about the resampling process for synthetic random effects. The resampling methods are the same as those from `syn_x_resample_inform`.
- **Whole Data Information**:

- `size`: An integer representing the total size of the combined original and synthetic datasets.
- `data`: A combined data frame of the original and synthetic datasets.
- `x`: A combined data frame of the original and synthetic predictor variables.
- `y`: A combined numeric vector of the original and synthetic response variables.
- `z`: A combined data frame of the original and synthetic random effect variables.
- `group`: A combined numeric vector representing the grouping variable for both original and synthetic datasets.

Examples

```
data(mtcars)
cat_init <- cat_lmm_initialization(
  formula = mpg ~ wt + (1 | cyl), # formula for simple model
  data = mtcars,
  x_cols = c("wt"), # Fixed effects
  y_col = "mpg", # Response variable
  z_cols = c("disp", "hp", "drat", "qsec", "vs", "am", "gear", "carb"), # Random effects
  group_col = "cyl", # Grouping column
  syn_size = 100, # Synthetic data size
  resample_by_group = FALSE, # Resampling option
  resample_only = FALSE, # Resampling method
  na_replace = mean # NA replacement method
)
cat_init
```

---

cat_lmm_tune *Catalytic Linear Mixed Model (LMM) Fitting Function by Tuning tau from a Sequence of tau Values*

Description

This function tunes a catalytic linear mixed model by performing cross-validation to estimate the optimal value of the tuning parameter tau. It finally uses the optimal tau value in the `lmer` function from the `lme4` package for model fitting. (Only consider one random effect variance)

Usage

```
cat_lmm_tune(cat_init, tau_seq = NULL, cross_validation_fold_num = 5)
```

Arguments

`cat_init`	A list generated from `cat_lmm_initialization`.
`tau_seq`	A numeric vector specifying the sequence of tau values to be tested. If NULL, a default sequence is generated based on the number of predictors.
`cross_validation_fold_num`	An integer representing the number of folds for cross-validation. Defaults to 5.

Value

A list containing the values of all the arguments and the following components:

`tau`	The optimal tau value determined from cross-validation.
`model`	The fitted lmer model object by using the optimal tau value.
`coefficients`	Coefficients of the fitted model by using the optimal tau value.
`risk_estimate_list`	Average prediction errors for each tau value.

Examples

```
data(mtcars)
cat_init <- cat_lmm_initialization(
  formula = mpg ~ wt + (1 | cyl), # formula for simple model
  data = mtcars,
  x_cols = c("wt"), # Fixed effects
  y_col = "mpg", # Response variable
  z_cols = c("disp", "hp", "drat", "qsec", "vs", "am", "gear", "carb"), # Random effects
  group_col = "cyl", # Grouping column
  syn_size = 100, # Synthetic data size
  resample_by_group = FALSE, # Resampling option
  resample_only = FALSE, # Resampling method
  na_replace = mean # NA replacement method
)

cat_model <- cat_lmm_tune(
  cat_init = cat_init, # Only accept object generated from cat_lmm_initialization
  tau_seq = c(1, 2), # Vector of weights for synthetic data
  cross_validation_fold_num = 3 # number of folds for cross-validation
)
cat_model
```

---

cross_validation *Perform Cross-Validation for Model Estimation*

Description

This function performs cross-validation for estimating risk over a sequence of tuning parameters (`tau_seq`) by fitting a Generalized Linear Model (GLM) to the data. It evaluates model performance by splitting the dataset into multiple folds, training the model on a subset of the data, and testing it on the remaining portion.

Usage

```
cross_validation(
  formula,
  cat_init,
```

```
  tau_seq,
  discrepancy_method,
  cross_validation_fold_num,
  ...
)
```

Arguments

`formula`	A formula specifying the GLMs. Should at least include response variables.
`cat_init`	A list generated from `cat_glm_initialization`.
`tau_seq`	A sequence of tuning parameter values (`tau`) over which cross-validation will be performed. Each value of `tau` is used to weight the synthetic data during model fitting.
`discrepancy_method`	A function used to calculate the discrepancy (error) between model predictions and actual values.
`cross_validation_fold_num`	The number of folds to use in cross-validation. The dataset will be randomly split into this number of subsets, and the model will be trained and tested on different combinations of these subsets.
`...`	Other arguments passed to other internal functions.

Details

1. **Randomization of the Data**: The data is randomly shuffled into `cross_validation_fold_num` subsets to ensure that the model is evaluated across different splits of the dataset.
2. **Model Training and Prediction**: For each fold, a training set is used to fit a GLM with varying values of `tau` (from `tau_seq`), and the model is evaluated on a test set. The training data consists of both the observed and synthetic data, with synthetic data weighted by `tau`.
3. **Risk Estimation**: After fitting the model, the `discrepancy_method` is used to calculate the prediction error for each combination of fold and `tau`. These errors are accumulated for each `tau`.
4. **Average Risk Estimate**: After completing all folds, the accumulated prediction errors are averaged over the number of folds to provide a final risk estimate for each value of `tau`.

Value

A numeric vector of averaged risk estimates, one for each value of `tau` in `tau_seq`.

cross_validation_cox *Perform Cross-Validation for Catalytic Cox Proportional-Hazards Model (COX) to Select Optimal tau*

Description

This function performs cross-validation for the catalytic Cox proportional-hazards model (COX) to estimate the likelihood associated with different values of tau. It splits the data into training and testing sets and computes prediction errors for model evaluation.

Usage

```
cross_validation_cox(
  formula,
  cat_init,
  method,
  tau_seq,
  cross_validation_fold_num,
  ...
)
```

Arguments

formula	A formula specifying the Cox model. Should at least include response variables.
cat_init	A list containing initialized parameters for the catalytic COX.
method	Character string specifying the optimization method used in the Cat-Cox model fitting.
tau_seq	A numeric vector of tau values for which to estimate likelihood.
cross_validation_fold_num	An integer indicating the number of folds for cross-validation.
...	Additional arguments passed to the cat_cox function for model fitting.

Value

A numeric vector containing the average likelihood estimates for each tau value.

cross_validation_lmm *Perform Cross-Validation for Catalytic Linear Mixed Model (LMM) to Select Optimal tau*

Description

This function performs cross-validation for the catalytic linear mixed model (LMM) to estimate the risk associated with different values of tau. It splits the data into training and testing sets and computes prediction errors for model evaluation.

Usage

```
cross_validation_lmm(cat_init, tau_seq, cross_validation_fold_num = 5)
```

Arguments

cat_init	A list containing initialized parameters for the catalytic LMM.
tau_seq	A numeric vector of tau values for which to estimate risk.
cross_validation_fold_num	An integer indicating the number of folds for cross-validation.

Value

A numeric vector containing the average risk estimates for each tau value.

---

extract_coefs *Extract and Format Model Coefficients*

Description

This function retrieves the coefficients from a x object, formats them with appropriate names, and rounds each coefficient to the specified number of decimal places. Optionally, the intercept can be included or excluded from the output.

Usage

```
extract_coefs(x, digit = 3)
```

Arguments

x	A model object generated from catalytic that containing model coefficients.
digit	An integer specifying the number of decimal places for rounding coefficients. Default is 3.

Value

A named numeric vector of model coefficients, rounded to the specified number of decimal places.

extract_dim *Extract Dimension Information from Model Initialization*

Description

This function retrieves and formats the dimensions of the dataset used in the model, including the number of observed and synthetic data points and the total number of rows and columns.

Usage

```
extract_dim(cat_init)
```

Arguments

cat_init	A list containing model initialization data, expected to include obs_size (observed data size), syn_size (synthetic data size), size (total data size), and x (the covariate matrix).

Value

A character string summarizing the dimensions of the dataset used in the model.

extract_stan_summary *Extract and Format Summary of Stan Model Results*

Description

This function extracts the summary statistics from a fitted Stan model stored within a x object, formats the parameter names, and rounds values to a specified number of decimal places. By default, the function includes an intercept term in the summary if present.

Usage

```
extract_stan_summary(x, digit = 3, with_intercept = TRUE)
```

Arguments

x	A model object generated from catalytic that containing a fitted Stan model.
digit	An integer specifying the number of decimal places to which the summary statistics should be rounded. Default is 3.
with_intercept	A logical value indicating whether the intercept should be included in the summary. If TRUE, the intercept is labeled and included in the formatted output. Default is TRUE.

Value

A matrix of rounded summary statistics from the Stan model, with row names representing parameter labels and columns containing summary values.

---

extract_tau_seq *Extract and Format Sequence of Tau Values*

Description

This function retrieves the sequence of tau values from a x object, rounds each value to the specified number of decimal places, and formats the output as a concise string. If the sequence contains more than 10 values, only the first 3 and last 3 values are shown, with ellipsis ("...") in between.

Usage

```
extract_tau_seq(x, digit = 3)
```

Arguments

x	A model object generated from catalytic that containing a sequence of tau values.
digit	An integer specifying the number of decimal places to which tau values should be rounded. Default is 3.

Value

A character string representing the rounded tau values, formatted for readability.

---

get_adjusted_cat_init *Adjusted Cat Initialization*

Description

This function adjusts the categorical initialization by creating a model frame for the predictors specified in the right-hand side of the formula and splits the adjusted data into observed and synthetic parts.

Usage

```
get_adjusted_cat_init(cat_init, formula_rhs)
```

Arguments

cat_init	The object generated from cat_glm_initialization, cat_cox_initialization or cat_lmm_initialization
formula_rhs	A formula specifying the right-hand side of the model for predictors.

Value

A list containing the original `cat_init` with added components: - `adj_x`: The adjusted model frame for the predictors. - `adj_obs_x`: The observed part of the adjusted predictors. - `adj_syn_x`: The synthetic part of the adjusted predictors.

`get_cox_gradient` *Compute the Gradient for Cox Proportional Hazards Model*

Description

This function computes the gradient for the Cox proportional hazards model. The gradient is calculated by considering the contributions of each observation to the gradient based on the risk set at each event time.

Usage

```
get_cox_gradient(X, time, status, coefs, entry_points)
```

Arguments

`X`	A matrix of covariates (design matrix) for the Cox model.
`time`	A numeric vector of event times.
`status`	A numeric vector of event indicators (1 for event, 0 for censored).
`coefs`	A numeric vector of coefficients for the Cox model.
`entry_points`	A numeric vector of entry times for the subjects. Defaults to 0.

Value

A numeric vector representing the gradient of the Cox proportional hazards model.

`get_cox_hessian` *Compute the Hessian Matrix for Cox Proportional Hazards Model*

Description

This function computes the Hessian matrix of the Cox proportional hazards model, which is used for estimating the covariance matrix of the coefficients. The Hessian is calculated by summing contributions from each event time in the risk set.

Usage

```
get_cox_hessian(X, time, status, coefs, entry_points)
```

Arguments

X	A matrix of covariates (design matrix) for the Cox model.
time	A numeric vector of event times.
status	A numeric vector of event indicators (1 for event, 0 for censored).
coefs	A numeric vector of coefficients for the Cox model.
entry_points	A numeric vector of entry times for the subjects. Defaults to 0.

Value

A matrix representing the negative Hessian of the Cox model.

---

get_cox_kappa *Estimate the kappa value for the synthetic Cox proportional hazards model*

Description

This function iterative estimates the kappa value for the synthetic Cox proportional hazards model using a vectorized approach for efficiency.

Usage

```
get_cox_kappa(X, time, status, hazard_constant)
```

Arguments

X	A matrix of covariates with rows representing observations and columns representing features.
time	A vector of time-to-event data.
status	A vector indicating event occurrence (1 = event, 0 = censored).
hazard_constant	A scalar representing the hazard constant. Defaults to NULL, in which case it's calculated internally.

Value

A numeric value representing the estimated kappa for the synthetic Cox model.

get_cox_partial_likelihood *Compute the Partial Likelihood for the Cox Proportional Hazards Model*

Description

This function calculates the partial likelihood for the Cox proportional hazards model. The partial likelihood is computed for the censored observations in the dataset.

Usage

```
get_cox_partial_likelihood(X, time, status, coefs, entry_points)
```

Arguments

X	A matrix of covariates with rows representing observations and columns representing features.
time	A vector of time-to-event data.
status	A vector indicating the event status (1 for event occurred, 0 for censored).
coefs	A vector of regression coefficients.
entry_points	A vector of entry points (optional). Defaults to NULL, in which case a vector of zeros is used.

Value

A numeric scalar representing the partial likelihood of the Cox model.

get_cox_qr_solve *Solve Linear System using QR Decomposition*

Description

This function solves the linear system defined by `hessian_matrix` and `gradient_vector` using QR decomposition. Any NA values in the resulting solution vector are replaced with 0.0001. If there is an error during the solution process, a vector of default values (0.0001) is returned instead.

Usage

```
get_cox_qr_solve(hessian_matrix, gradient_vector)
```

Arguments

hessian_matrix	A matrix of coefficients representing the system of linear equations.
gradient_vector	A numeric vector representing the constants in the linear system.

Value

A numeric vector representing the solution to the linear system. NA values in the solution are replaced with a small value (0.0001). If an error occurs during solving, a vector of default values (0.0001) is returned.

`get_cox_risk_and_failure_sets`
Calculate Risk and Failure Sets for Cox Proportional Hazards Model

---

Description

This function calculates the risk and failure sets for subjects in a Cox proportional hazards model based on their time-to-event data, status, and an indicator vector.

Usage

```
get_cox_risk_and_failure_sets(time_vector, status_vector, indicator_vector)
```

Arguments

`time_vector`	A numeric vector of time-to-event data for each subject.
`status_vector`	A numeric vector indicating event occurrence (1 = event, 0 = censored).
`indicator_vector`	A numeric vector representing the indicator times used to define risk and failure sets.

Value

A list containing two elements:

- `risk_set`: A matrix indicating which subjects are at risk at each time point.
- `failure_set`: A matrix indicating which subjects experienced an event at each time point.

`get_cox_risk_set_idx` *Identify the risk set indices for Cox proportional hazards model*

Description

This function returns the indices of the risk set for a given time of interest in the Cox proportional hazards model.

Usage

```
get_cox_risk_set_idx(
  time_of_interest,
  entry_vector,
  time_vector,
  status_vector
)
```

Arguments

time_of_interest	A numeric value representing the time at which the risk set is calculated.
entry_vector	A numeric vector representing the entry times of subjects.
time_vector	A numeric vector representing the time-to-event or censoring times of subjects.
status_vector	A numeric vector indicating event occurrence (1) or censoring (0) for each subject.

Value

A vector of indices representing the subjects at risk at the specified time of interest.

---

get_cox_syn_gradient *Compute the gradient of the synthetic Cox proportional hazards model*

Description

This function calculates the gradient of the synthetic Cox proportional hazards model using a vectorized approach.

Usage

```
get_cox_syn_gradient(X, time, coefs, hazard_constant)
```

Arguments

X	A matrix of covariates with rows representing observations and columns representing features.
time	A vector of time-to-event data.
coefs	A vector of regression coefficients.
hazard_constant	A scalar representing the hazard constant.

Value

A numeric vector representing the gradient of the synthetic Cox model.

get_cox_syn_hessian *Compute the Synthetic Hessian Matrix for Cox Proportional Hazards Model*

Description

This function computes the synthetic Hessian matrix for the Cox proportional hazards model. The Hessian is calculated by summing the contributions from each individual observation, scaled by the hazard constant and the time of the event.

Usage

```
get_cox_syn_hessian(X, time, coefs, hazard_constant)
```

Arguments

X	A matrix of covariates (design matrix) for the Cox model.
time	A numeric vector of event times.
coefs	A numeric vector of coefficients for the Cox model.
hazard_constant	A numeric value representing the hazard constant.

Value

A matrix representing the synthetic Hessian of the Cox model.

get_discrepancy *Compute Discrepancy Measures*

Description

This function computes various discrepancy measures between observed and estimated values. It supports different methods including logarithmic error, square error, classification error, and logistic deviance.

Usage

```
get_discrepancy(
  discrepancy_method = c("mean_logarithmic_error", "mean_square_error",
    "mean_classification_error", "logistic_deviance"),
  family_string = NULL,
  X = NULL,
  Y = NULL,
  coefs = NULL,
  est_Y = NULL
)
```

Arguments

discrepancy_method	A character string specifying the discrepancy method to use. Options are: **"logarithmic_error"** Logarithmic error, suitable for probabilities. **"mean_square_error"** Mean squared error. **"mean_classification_error"** Mean of classification error, suitable for binary outcomes. **"logistic_deviance"** Logistic deviance, computed using a GLM model.
family_string	A GLM family in string (e.g., "binomial") used to compute logistic deviance.
X	A matrix of predictor variables.
Y	A vector or data frame of observed values.
coefs	A vector of coefficients for the GLM model.
est_Y	A vector of estimated values. If not provided, it will be computed using get_glm_mean with the specified family.

Value

A numeric value representing the discrepancy between observed and estimated values.

---

get_formula_lhs *Extract Left-Hand Side of Formula as String*

Description

This function extracts the left-hand side (LHS) of a formula object and converts it to a character string. It uses get_formula_string to ensure consistent formatting.

Usage

```
get_formula_lhs(formula)
```

Arguments

formula	A formula object from which the LHS will be extracted.

Value

A character string representing the left-hand side of the formula.

get_formula_rhs *Extract the Right-Hand Side of a Formula*

Description

This function extracts the right-hand side (RHS) of a formula and returns it as a character string. Optionally, it can include a tilde (~) at the beginning of the RHS.

Usage

```
get_formula_rhs(formula, with_tilde = FALSE)
```

Arguments

formula	A formula object from which to extract the RHS.
with_tilde	Logical, indicating whether to include a tilde (~) at the beginning of the RHS. Defaults to FALSE.

Value

A character string representing the right-hand side of the formula. If with_tilde is TRUE, the string includes a leading tilde.

get_formula_string *Convert Formula to String*

Description

This function converts a formula object to a character string. It removes extra whitespace and formats the formula as a single line.

Usage

```
get_formula_string(formula)
```

Arguments

formula	A formula object to be converted to a string.

Value

A character string representing the formula.

get_glm_custom_var *Get Custom Variance for Generalized Linear Model (GLM)*

Description

This function calculates a custom variance for a Generalized Linear Model (GLM) based on the specified formula, the model initialization object, and a scaling factor `tau`. The custom variance is computed by adjusting the residuals of the fitted model and returning a weighted sum of squared residuals.

Usage

```
get_glm_custom_var(formula, cat_init, tau)
```

Arguments

`formula`	A formula object specifying the GLM model to be fitted, such as `response ~ predictors`.
`cat_init`	A list object containing the initialization data for the model. Generated from `cat_initialization`
`tau`	A numeric value representing a scaling factor for down-weighting synthetic data

Value

A numeric value representing the custom variance for the GLM model.

get_glm_diag_approx_cov *Compute Diagonal Approximate Covariance Matrix*

Description

This function computes the diagonal elements of the approximate covariance matrix for the coefficients in a generalized linear model (GLM). The covariance is derived from the second derivative (Hessian) of the log-likelihood function.

Usage

```
get_glm_diag_approx_cov(X, model)
```

Arguments

`X`	Matrix. The design matrix (predictors) for the GLM.
`model`	A fitted GLM model object. The object should contain the fitted values and prior weights necessary for computing the Hessian.

Value

Numeric vector. The diagonal elements of the approximate covariance matrix.

---

get_glm_family_string *Retrieve GLM Family Name or Name with Link Function*

Description

This function retrieves the name of a GLM family or, optionally, the family name with the associated link function.

Usage

```
get_glm_family_string(family, with_link = FALSE)
```

Arguments

family	Character or function. The name of the GLM family (as a string) or a function that returns a GLM family object.
with_link	Logical. If TRUE, returns the family name along with the link function in the format "family \[link\]". If FALSE, only the family name is returned. Default is FALSE.

Value

A character string. The name of the GLM family, or the name with the link function if with_link is TRUE.

---

get_glm_lambda *Compute Lambda Based on Discrepancy Method*

Description

This function calculates a lambda value based on the selected discrepancy method for a generalized linear model (GLM). The discrepancy method determines the type of error or deviance used in the calculation.

Usage

```
get_glm_lambda(
  discrepancy_method = c("mean_square_error", "mean_classification_error",
    "logistic_deviance"),
  X,
  coefs
)
```

Arguments

discrepancy_method	Character. A string specifying the type of discrepancy method to use. Options are "mean_square_error", "mean_classification_error", or "logistic_deviance". Default is "mean_square_error".
X	Matrix. The design matrix (predictors) for the GLM.
coefs	Numeric vector. The coefficients for the GLM.

Value

Numeric. The computed lambda value based on the selected discrepancy method.

---

get_glm_log_density *Compute Log Density Based on GLM Family*

Description

This function calculates the log density of the response variable given a generalized linear model (GLM) based on the specified family. The log density is computed differently for binomial and gaussian families.

Usage

```
get_glm_log_density(family_string, X, Y, coefs, weights = 1)
```

Arguments

family_string	Character. The GLM family to use. Options are "binomial" or "gaussian".
X	Matrix. The design matrix (predictors) for the GLM.
Y	Vector or data frame. The response variable for the GLM. If a data frame, it is converted to a numeric vector.
coefs	Numeric vector. The coefficients for the GLM.
weights	Numeric vector. Weights for the observations. Default is 1 (no weighting).

Value

Numeric. The computed log density of the response variable based on the specified family.

get_glm_log_density_grad *Compute Gradient of Log Density for GLM Families*

Description

This function calculates the gradient of the log density with respect to the coefficients for a given GLM family based on the provided predictors, response variable, and weights.

Usage

```
get_glm_log_density_grad(family_string, X, Y, coefs, weights = 1)
```

Arguments

family_string	Character. The GLM family to use. Options are "binomial" or "gaussian".
X	Matrix. The design matrix (predictors) for the GLM.
Y	Vector. The response variable.
coefs	Numeric vector. The coefficients for the GLM.
weights	Numeric vector. The weights for the GLM. Default is 1.

Value

Numeric vector. The gradient of the log density with respect to the coefficients

get_glm_mean *Compute Mean Based on GLM Family*

Description

This function calculates the mean of the response variable for a generalized linear model (GLM) based on the specified family. The calculation depends on whether the family is binomial or gaussian.

Usage

```
get_glm_mean(family_string, X, coefs)
```

Arguments

family_string	Character. The GLM family to use. Options are "binomial" or "gaussian".
X	Matrix. The design matrix (predictors) for the GLM.
coefs	Numeric vector. The coefficients for the GLM.

Value

Numeric vector. The computed mean of the response variable based on the specified family.

get_glm_sample_data *Generate Sample Data for GLM*

Description

This function generates sample data for a specified GLM family. It can generate binomial or Gaussian distributed data based on the provided parameters.

Usage

```
get_glm_sample_data(family_string, n = 10, mean = 0, sd = 1)
```

Arguments

family_string	Character. The family of the GLM. Options are "binomial" or "gaussian".
n	Integer. The number of samples to generate.
mean	Numeric. The mean of the distribution (used for both binomial and Gaussian).
sd	Numeric. The standard deviation of the distribution (used only for Gaussian).

Value

Numeric vector of generated sample data.

get_hmc_mcmc_result *Run Hamiltonian Monte Carlo to Get MCMC Sample Result*

Description

This function uses Hamiltonian Monte Carlo (HMC) to generate samples for Markov Chain Monte Carlo (MCMC) sampling from a target distribution specified by neg_log_den_func. Each iteration performs a full HMC update to generate a new sample position.

Usage

```
get_hmc_mcmc_result(
  neg_log_den_func,
  neg_log_den_grad_func,
  coefs_0,
  iter = 5,
  hmc_scale = 0.01,
  hmc_steps = 5
)
```

Arguments

neg_log_den_func	A function that computes the negative log-density of the target distribution at a given position.
neg_log_den_grad_func	A function that computes the gradient of neg_log_den_func at a given position.
coefs_0	A numeric vector specifying the initial position of the chain in the parameter space.
iter	An integer specifying the number of HMC sampling iterations. Defaults to 5.
hmc_scale	A numeric value representing the scale factor for the leapfrog step size in the HMC update. Defaults to 0.01.
hmc_steps	An integer specifying the number of leapfrog steps in each HMC update. Defaults to 5.

Value

A numeric vector representing the final position in the parameter space after the specified number of iterations.

---

get_linear_predictor *Compute Linear Predictor*

Description

This function computes the linear predictor from a matrix of predictor variables and a vector of coefficients. It handles cases with and without an intercept term.

Usage

```
get_linear_predictor(X, coefs)
```

Arguments

X	A matrix of predictor variables.
coefs	A vector of coefficients. It should be either the same length as the number of columns in X (for models without an intercept) or one more than the number of columns in X (for models with an intercept).

Value

A vector of linear predictor values.

`get_resampled_df` *Resampling Methods for Data Processing*

Description

This function includes various resampling methods applied to input data for each column to prepare it for analysis. These methods help to transform the data distribution and improve model fitting.

Usage

```
get_resampled_df(
  data,
  resample_size,
  data_degree = NULL,
  resample_only = FALSE
)
```

Arguments

`data`	A data frame to be resampled.
`resample_size`	An integer specifying the size of the resample.
`data_degree`	A numeric vector indicating the degree of each column in the data (optional).
`resample_only`	A logical value indicating whether to return only the resampled data (default is FALSE).

Details

- **Coordinate**: This method refers to the preservation of the original data values as reference coordinates during processing. It ensures that the transformations applied are based on the initial structure of the data.
- **Deskewing**:Deskewing is the process of adjusting the data distribution to reduce skewness, making it more symmetric. If the absolute value of skewness is greater than or equal to 1, deskewing techniques will be applied to normalize the distribution, which can enhance model performance.
- **Smoothing**: Smoothing techniques reduce noise in the data by averaging or modifying data points. This is especially useful when there are many unique values in the original data column, as it helps to stabilize the dataset and prevent overfitting during model training.
- **Flattening**: Flattening modifies the data to create a more uniform distribution across its range. This method is employed when the frequency of certain categories in categorical variables is low, replacing some original values with randomly selected unique values from the dataset to reduce sparsity.
- **Symmetrizing**: Symmetrizing adjusts the data so that it becomes more balanced around its mean. This is crucial for achieving better statistical properties and improving the robustness of the model fitting process.

Value

A list containing:

`resampled_df` A data frame of resampled data.

`resampled_df_log`
A data frame recording the resampling process for each column.

`get_standardized_data` *Standardize Data*

Description

This function standardizes a dataset by converting columns to numeric or factor types and replacing NA values. For continuous variables, NA values are replaced with either a specific numeric value or a computed statistic. For categorical variables, NA values are replaced with the mode of the column.

Usage

```
get_standardized_data(data, na_replace = stats::na.omit)
```

Arguments

`data`	A data frame to be standardized.
`na_replace`	A function or numeric value used to replace NA values. If a function, it should take a vector and return a replacement value. If a numeric value, it is used directly to replace NA values in continuous columns. The default is `stats::na.omit`, which omits rows with NA values (used as an indicator here, not the actual replacement value).

Value

A data frame where columns have been converted to numeric or factor types, and NA values have been replaced according to the method specified.

`get_stan_model` *Generate Stan Model Based on Specified Parameters*

Description

This function retrieves a Stan model file based on a combination of input parameters, constructs the file path, and loads the Stan model.

Usage

```
get_stan_model(
  type = c("glm", "cox"),
  glm_family_string = c("gaussian", "binomial"),
  joint_tau = FALSE,
  glm_binomial_joint_theta = FALSE,
  glm_binomial_joint_alpha = FALSE,
  glm_gaussian_known_variance = FALSE
)
```

Arguments

`type`	Character, either `"glm"` or `"cox"`, specifying the type of model to load.
`glm_family_string`	Character, specifying the family for GLM models, either `"binomial"` or `"gaussian"`. Required for `"glm"` models, ignored for `"cox"` models.
`joint_tau`	Logical, if `TRUE`, includes "joint" in the file name to indicate a joint model with tau parameter.
`glm_binomial_joint_theta`	Logical, if `TRUE` and `glm_family_string` is `"binomial"`, includes "theta" in the file name for joint theta parameter.
`glm_binomial_joint_alpha`	Logical, if `TRUE` and `glm_family_string` is `"binomial"`, includes "alpha" in the file name for joint alpha parameter.
`glm_gaussian_known_variance`	Logical, if `TRUE` and `glm_family_string` is `"gaussian"`, includes "known_variance" in the file name to specify known variance.

Value

A compiled Stan model loaded by `rstan::stan_model`.

`hmc_neal_2010` *Hamiltonian Monte Carlo (HMC) Implementation*

Description

This function implements the Hamiltonian Monte Carlo algorithm as described by Radford M. Neal (2010) in "MCMC using Hamiltonian dynamics", which is a part of the Handbook of Markov Chain Monte Carlo. The method uses Hamiltonian dynamics to propose new positions and then applies the Metropolis criterion to decide whether to accept or reject the new position.

Usage

```
hmc_neal_2010(
  neg_log_den_func,
  neg_log_den_grad_func,
  leapfrog_stepsize,
  leapfrog_step,
  current_pos
)
```

Arguments

`neg_log_den_func`	A function that evaluates the negative log of the density (potential energy) of the distribution to be sampled, including any constants.
`neg_log_den_grad_func`	A function that computes the gradient of `neg_log_den_func`.
`leapfrog_stepsize`	A numeric value specifying the step size for the leapfrog integration method.
`leapfrog_step`	A numeric value specifying the number of leapfrog steps to take to propose a new state.
`current_pos`	A numeric vector representing the current position (state) of the system.

Details

This function was written for illustrative purposes. More elaborate on Radford M. Neal's personal webpage (http://www.cs.utoronto.ca/~radford/).

Value

A list containing the following elements:

- `position`: The position of the system after the leapfrog steps, which is the proposed new position if accepted, or the current position if rejected.
- `potential_energy`: The potential energy of the proposed position.
- `accepted`: A logical value indicating whether the proposal was accepted (TRUE) or rejected (FALSE).

References

Neal, R. M. (2012). MCMC using Hamiltonian dynamics. arXiv:1206.1901. Available at: https://arxiv.org/pdf/1206.1901

`is.continuous` *Check if a Variable is Continuous*

Description

This function checks whether a given vector represents a continuous variable. A continuous variable is numeric and has more than two unique values.

Usage

```
is.continuous(lst)
```

Arguments

`lst`	A vector to be checked.

Value

A logical value indicating whether the input vector is considered continuous. Returns `TRUE` if the vector is numeric and has more than two unique values; otherwise, returns `FALSE`.

`mallowian_estimate` *Perform Mallowian Estimate for Model Risk (Only Applicable for Gaussian Family)*

Description

This function calculates the Mallowian estimate for model risk by fitting a sequence of Generalized Linear Models (GLMs) with varying values of `tau`. It uses the in-sample prediction error along with a regularized projection matrix to estimate the model risk. The `tau` parameter influences the weighting of synthetic data during model fitting.

Usage

```
mallowian_estimate(formula, cat_init, tau_seq, ...)
```

Arguments

formula	A formula specifying the GLMs. Should at least include response variables.
cat_init	A list generated from cat_glm_initialization.
tau_seq	A sequence of tuning parameter values (tau) over which the Mallowian estimate will be computed. Each value of tau is used to weight the synthetic data during model fitting.
...	Other arguments passed to other internal functions.

Details

1. **Model Fitting**: For each value of tau in tau_seq, the function fits a GLM model using the observed and synthetic data. The synthetic data is weighted by the corresponding tau value during the fitting process.
2. **In-sample Prediction Error**: After fitting the model, the function computes the in-sample prediction error (Mean Squared Error) to assess the model's performance.
3. **Regularized Projection Matrix**: The function calculates a regularized projection matrix using the observed and synthetic data, which influences the covariance matrix used in risk estimation.
4. **Mallowian Risk Estimate**: The final Mallowian risk estimate is computed by combining the in-sample prediction error with a penalty term involving the projection matrix and a variance term. This estimate is calculated for each value of tau in tau_seq.

Value

A numeric vector of Mallowian risk estimates, one for each value of tau in tau_seq.

---

parametric_bootstrap *Perform Parametric Bootstrap for Model Risk Estimation*

Description

This function performs parametric bootstrapping to estimate model risk. It fits a sequence of Generalized Linear Models (GLMs) with different values of tau, calculates the in-sample prediction error, and incorporates deviations from the bootstrap response samples. The final risk estimate is obtained by combining the in-sample error and the covariance penalty derived from the bootstrap samples.

Usage

```
parametric_bootstrap(
  formula,
  cat_init,
  tau_seq,
  tau_0,
  discrepancy_method,
```

```
  parametric_bootstrap_iteration_times,
  ...
)
```

Arguments

`formula`	A formula specifying the GLMs. Should at least include response variables.
`cat_init`	A list generated from `cat_glm_initialization`.
`tau_seq`	A sequence of tuning parameter values (`tau`) over which the model risk will be estimated. Each `tau` value is used to weight the synthetic data during model fitting.
`tau_0`	A reference value for `tau` used in the preliminary estimate model and variance calculation.
`discrepancy_method`	The method used to calculate the discrepancy (e.g., logistic deviance).
`parametric_bootstrap_iteration_times`	The number of bootstrap iterations to perform.
`...`	Other arguments passed to other internal functions.

Details

1. **Preliminary Estimate Model**: The function first fits a GLM model using the observed and synthetic data with an initial value of `tau_0` for the synthetic data weights.
2. **Bootstrap Samples**: The function generates bootstrap response samples based on the mean and standard deviation of the preliminary estimate model, using parametric bootstrapping.
3. **In-sample Prediction Error**: For each value of `tau` in `tau_seq`, the function computes the in-sample prediction error (e.g., using logistic deviance).
4. **Bootstrap Models**: For each bootstrap iteration, the function fits a GLM using the bootstrap response samples and calculates the corresponding lambda values.
5. **Covariance Penalty**: The function approximates the covariance penalty using the weighted deviations across all bootstrap iterations.
6. **Final Risk Estimate**: The final model risk estimate is calculated by summing the in-sample prediction error and the average weighted deviations from the bootstrap response samples.

Value

A numeric vector containing the risk estimates for each `tau` in `tau_seq`.

plot.cat_tune *Plot Likelihood or Risk Estimate vs. Tau for Tuning Model*

Description

This function generates a plot showing the relationship between the tuning parameter tau and either the likelihood score (for a cat_cox_tune model) or the risk estimate (for other models) during cross-validation or other model evaluation methods. The plot highlights the optimal tau value and provides visual cues for the best tuning parameter based on the specified method.

Usage

```
## S3 method for class 'cat_tune'
plot(x, digit = 2, legend_pos = "topright", text_pos = 3, ...)
```

Arguments

x	A fitted model object of class cat_tune that contains the results of the tuning process. This object includes the likelihood or risk estimate lists, the tuning sequence (tau_seq), and the selected optimal tau.
digit	An integer specifying the number of decimal places to round the displayed values (default is 2).
legend_pos	A character string specifying the position of the legend on the plot (default is "topright").
text_pos	An integer specifying the position of the text label on the plot (default is 3, which places the text above the point).
...	Additional parameters to pass to other functions.

Details

The function generates a line plot with tau_seq on the x-axis and either the likelihood score or risk estimate on the y-axis. If the model is of class cat_cox_tune, the plot shows the likelihood score, while for other models, it shows the risk estimates. The optimal tau is marked with a red cross, and red dashed lines are drawn to highlight the optimal point on the plot.

Value

A plot with the specified y-values plotted against tau_seq, including a highlighted optimal tau point.

predict.cat_cox *Predict Linear Predictor for New Data Using a Fitted Cox Model*

Description

This function calculates the linear predictor (LP) for new data points based on a fitted Cox proportional hazards model.

Usage

```
## S3 method for class 'cat_cox'
predict(object, newdata = NULL, ...)
```

Arguments

object	A fitted model object of class cat_cox, containing the COX fit and model details.
newdata	An optional data frame with new predictor values. If NULL, the function uses the observation data from the model's initialization object.
...	Additional arguments passed to other functions.

Value

A vector of linear predictor values for the specified new data.

predict.cat_glm *Predict Outcome for New Data Using a Fitted GLM Model*

Description

This function generates predictions for new data points based on a fitted categorical Generalized Linear Model (GLM) object. Depending on the type of model, it either uses stats::predict.glm or calculates predictions based on the model coefficients.

Usage

```
## S3 method for class 'cat_glm'
predict(object, newdata = NULL, ...)
```

Arguments

object	A fitted model object of class cat_glm, containing the GLM fit and model details.
newdata	An optional data frame containing new predictor values. If NULL, the function uses the observation data from the model's initialization object.
...	Additional arguments passed to stats::predict.glm, if applicable. User should input type = c("link", "response", "terms") for different regression models.

Value

A vector of predicted values for the specified new data.

---

predict.cat_lmm *Predict Linear Predictor for New Data Using a Fitted Linear Mixed Model*

Description

This function calculates the linear predictor (LP) for new data points based on a fitted linear mixed model (LMM) stored in object.

Usage

```
## S3 method for class 'cat_lmm'
predict(object, newdata = NULL, ...)
```

Arguments

object	A fitted model object of class cat_lmm, containing the LMM fit and model details.
newdata	An optional data frame with new predictor values. If NULL, the function uses the observation data from the model's initialization object.
...	Additional arguments passed to other functions.

Value

A vector of linear predictor values for the specified new data.

print.cat *Print Method for* cat *Object*

Description

The print.cat function provides a detailed summary of the cat object, displaying key information about the model and its settings, including model type, covariates, formula, tau values, and relevant coefficients.

Usage

```
## S3 method for class 'cat'
print(x, digit = 3, detail = TRUE, ...)
```

Arguments

x	An object of class cat, representing a fitted model.
digit	An integer indicating the number of decimal places for printing coefficient estimates. Default is 3.
detail	A logical indicating whether to print additional details for interpreting the output. Default is TRUE.
...	Additional parameters to pass to other functions.

Details

This function customizes the output based on the model type stored within the x object, such as GLM, Cox, or other types of models.

The print.cat function prints a summary of the model stored in the x object. It will display different information depending on the model's type (GLM, Cox, etc.). It will show:

- The model's function name.
- The dimensions of the covariates used in the model.
- The tau values.
- Model-specific details such as family for GLMs or method and iteration info for Cox models.
- Coefficients related to the model.

Value

The x object is returned invisibly.

print.cat_bayes *Print Summary of* cat_bayes *Model*

Description

This function prints a formatted summary of a cat_bayes model object, displaying key parameters and settings of the fitted model, including the formula, covariate dimensions, tau (if applicable), family, and algorithm settings, as well as the coefficients' summary.

Usage

```
## S3 method for class 'cat_bayes'
print(x, digit = 3, detail = TRUE, ...)
```

Arguments

x	An object of class cat_tune, typically resulting from a tuning process, including cat_glm_bayes, cat_glm_bayes_joint, cat_cox_bayes and cat_cox_bayes_joint.
digit	An integer indicating the number of decimal places for printing coefficient estimates. Default is 3.
detail	A logical value indicating whether to include additional detailed output at the end of the summary. If TRUE, it will print additional interpretation help.
...	Additional parameters to pass to other functions.

Details

This function provides an organized printout of essential details from a Bayesian model fit. It includes the model formula, dimensionality of covariates, model family, Stan algorithm settings, and summary of the model coefficients. If detail is set to TRUE, additional information on interpreting the output is included.

Value

The x object is returned invisibly.

print.cat_gibbs *Print Summary of* cat_gibbs *Model*

Description

This function prints a summary of the cat_gibbs model, displaying details about the formula, covariate dimensions, family, coefficients, and Gibbs sampling settings.

Usage

```
## S3 method for class 'cat_gibbs'
print(x, digit = 3, detail = TRUE, ...)
```

Arguments

x	A cat_gibbs model object containing the results of a Bayesian GLM fitted using Gibbs sampling.
digit	An integer indicating the number of decimal places for printing coefficient estimates. Default is 3.
detail	A logical value indicating whether to include additional detailed output at the end of the summary. If TRUE, it will print additional interpretation help.
...	Additional parameters to pass to other functions.

Details

The summary includes:

- The function name and formula used in the model.
- Dimensions of the covariate matrix.
- Family and link function details.
- Sampling information, including the total iterations, warm-up iterations, and effective Gibbs sampling post-warmup.
- Coefficients with summary statistics and effective sample size.

If detail is set to TRUE, additional guidance for interpreting the printed output is provided.

Value

The x object is returned invisibly.

---

print.cat_initialization *Print Summary for Catalytic Initialization Model*

Description

This function provides a comprehensive summary of a catalytic initialization model object (cat_init), including formula details, data dimensions, and sample data previews.

Usage

```
## S3 method for class 'cat_initialization'
print(x, show_data = TRUE, detail = TRUE, ...)
```

Arguments

x	A catalytic initialization model object containing formula, family, data dimensions, and sampling details.
show_data	Logical, default TRUE. If TRUE, previews the head of both observation and synthetic data (up to the first 10 columns).
detail	Logical, default TRUE. If TRUE, adds guidance for interpreting the output.
...	Additional parameters to pass to other functions.

Details

The function provides a detailed overview of the initialization process for the cat_initialization model, including:

- The formula used in the model.
- The type of model (if Gaussian, the known or unknown variance is specified).
- The family of the Generalized Linear Model (GLM), along with the associated link function.
- The dimensions of the observation and synthetic data sets, with an option to display the first few rows.
- Information on the data generation process if available.

The show_data parameter controls whether the first few rows of the data are printed, while the detail parameter controls whether additional help for interpreting the printed output is displayed.

Value

Invisibly returns the x object.

---

print.cat_tune *Print Method for* cat_tune *Object*

Description

This function prints a summary of the cat_tune object, displaying key details such as the function name, dimensions of covariates, tau sequence, optimal tau, likelihood or risk estimate, and the model's coefficients.

Usage

```
## S3 method for class 'cat_tune'
print(x, digit = 3, detail = TRUE, ...)
```

Arguments

x	An object of class cat_tune, typically resulting from a tuning process, including cat_glm_tune, cat_cox_tune and cat_lmm_tune.
digit	An integer indicating the number of decimal places for printing coefficient estimates. Default is 3.
detail	A logical value indicating whether to include additional detailed output at the end of the summary. If TRUE, it will print additional interpretation help.
...	Additional parameters to pass to other functions.

Details

This method provides a comprehensive overview of the tuning process for the model, including the tau sequence and optimal tau, along with either the maximum likelihood (for Cox models) or minimum risk estimate (for other models). It also displays the coefficients of the model.

The function also checks if the x is a Cox model (cat_cox_tune) to adjust the interpretation of the output.

Value

The x object is returned invisibly.

---

print_df_head_tail *Print Data Frame with Head and Tail Rows*

Description

This function displays the first 5 and last 5 rows of a data frame. Column names are displayed only for the first 5 rows, with ellipses (. . .) in the middle to indicate additional rows.

Usage

```
print_df_head_tail(df, digit = 3)
```

Arguments

df	A data frame to display.
digit	An integer specifying the number of decimal places to which the summary statistics should be rounded. Default is 3.

Value

Invisibly returns the original data frame.

print_glm_bayes_joint_binomial_suggestion
Generate Suggestions for Bayesian Joint Binomial GLM Parameter Estimation

Description

This function provides suggestions for improving the parameter estimation process in Bayesian joint Binomial GLM modeling based on the diagnostic output from a Stan model. It evaluates the results and suggests adjustments to improve model fit and convergence.

Usage

```
print_glm_bayes_joint_binomial_suggestion(
  alpha,
  stan_iter,
  stan_sample_model,
  binomial_joint_theta,
  binomial_joint_alpha,
  binomial_tau_lower
)
```

Arguments

alpha	Numeric. The alpha parameter used in the prior distribution.
stan_iter	Integer. The number of iterations used in Stan sampling.
stan_sample_model	Stan model object containing sampling results.
binomial_joint_theta	Logical. Whether to use theta in the binomial model.
binomial_joint_alpha	Logical. Whether to use joint alpha in the binomial model.
binomial_tau_lower	Numeric. The lower bound for tau in the binomial model.

Value

NULL. The function prints suggestions to the console based on the model diagnostics.

`steinian_estimate` *Perform Steinian Estimate for Model Risk (Only Applicable for Binomial Family)*

Description

This function computes the Steinian estimate for model risk by fitting a sequence of Generalized Linear Models (GLMs) with varying values of `tau`. It combines the preliminary estimate from a model fitted with an initial `tau_0` value with a penalty term that incorporates the in-sample prediction error and a covariance penalty, which is based on models fitted by inverting the response of individual observations.

Usage

```
steinian_estimate(formula, cat_init, tau_seq, tau_0, ...)
```

Arguments

`formula`	A formula specifying the GLMs. Should at least include response variables.
`cat_init`	A list generated from `cat_glm_initialization`.
`tau_seq`	A sequence of tuning parameter values (`tau`) over which the Steinian estimate will be computed. Each value of `tau` is used to weight the synthetic data during model fitting.
`tau_0`	A reference value for `tau` that is used in the calculation of the preliminary estimate model and the variance term.
`...`	Other arguments passed to other internal functions.

Details

1. **Preliminary Estimate Model**: The function first fits a GLM model using the observed and synthetic data with an initial value of `tau_0` for the synthetic data weights.
2. **In-sample Prediction Error**: For each value of `tau` in `tau_seq`, the function computes the in-sample prediction error (logistic deviance).
3. **Steinian Penalty**: The function calculates the Steinian covariance penalty for each observation by fitting a modified model that inverts one observation at a time. The penalty is added to the in-sample prediction error to obtain the final risk estimate.
4. **Steinian Risk Estimate**: The final Steinian risk estimate is calculated by summing the in-sample prediction error and the Steinian penalty term for each value of `tau` in `tau_seq`.

Value

A numeric vector of Steinian risk estimates, one for each value of `tau` in `tau_seq`.

swim *Simulated SWIM Dataset with Binary Response*

Description

Simulated SWIM Dataset with Binary Response

Usage

```
data(swim)
```

Format

A list containing the following elements:

x A 3211 by 12 matrix of numeric values.

female Binary variable indicating gender (1 = Female, 0 = Male).

agege35 Binary variable indicating if the individual is aged 35 or older (1 = Age 35 or older, 0 = Younger than 35).

hsdip Binary variable indicating if the individual has a high school diploma (1 = Has diploma, 0 = No diploma).

nevmar Binary variable indicating if the individual has never been married (1 = Never married, 0 = Ever married).

divwid Binary variable indicating if the individual is divorced or widowed (1 = Divorced or widowed, 0 = Otherwise).

numchild Numerical variable indicating the number of children the individual has.

childlt6 Binary variable indicating if the individual has children under the age of 6 (1 = Has children under 6, 0 = No children under 6).

blknh Binary variable indicating if the individual is black non-Hispanic (1 = Black Non-Hispanic, 0 = Otherwise).

hisp Binary variable indicating if the individual is Hispanic (1 = Newly Hispanic, 0 = Otherwise).

earnyrm1 Numerical variable indicating the individual's earnings one year prior to the study (possibly negative earnings or debt).

empyrm1 Binary variable indicating if the individual was employed one year prior to the study (1 = Employed, 0 = Unemployed).

enrol Binary variable indicating if the individual was enrolled the job hunting training session (1 = Enrolled, 0 = Unenrolled).

y A 3211 by 1 matrix containing zeros and ones.

empyr1 Binary variable indicating if the individual was employed one year after the start of the study (1 = Employed, 0 = Unemployed).

Note

The dataset used in this study was simulated based on the patterns and results described in "The Saturation Work Initiative Model in San Diego: A Five-Year Follow-up Study". This data is not directly extracted from the book but was generated to emulate a similar structure for research and educational purposes.

References

Friedlander, D., & Hamilton, G. (1993). The Saturation Work Initiative Model in San Diego: A Five-Year Follow-up Study.

`traceplot` *Traceplot for Bayesian Model Sampling*

Description

The `traceplot` function is a generic function used to generate traceplots for Bayesian model sampling, primarily for assessing the convergence and mixing of Markov Chain Monte Carlo (MCMC) chains. This function dispatches specific traceplot methods depending on the class of the `object` object.

Usage

```
traceplot(object, ...)
```

Arguments

`object`	An object representing a Bayesian model, typically generated by the `cat_glm_bayes` or `cat_cox_bayes` functions, or similar models with Bayesian sampling results. The function uses S3 method dispatch to apply the appropriate `traceplot` method based on the class of `object`.
`...`	Additional arguments passed to specific `traceplot` methods for customization, such as selecting parameters to plot or setting display options.

Details

This generic `traceplot` function allows for flexible visualization of MCMC chains across different types of Bayesian models. Specific `traceplot` methods, such as `traceplot.cat_bayes`, are dispatched based on the `object` class to produce tailored traceplots, providing insights into the sampling progress and convergence diagnostics of each chain.

Value

A traceplot displaying the MCMC sampling chains for each parameter, assisting in convergence analysis. The exact output format depends on the specific `traceplot` method applied.

traceplot.cat_bayes *Traceplot for Bayesian Sampling Model*

Description

This function generates a traceplot for the Bayesian sampling model fitted using `rstan`. It utilizes the `traceplot` function from the `rstan` package to visualize the sampling progress and convergence of the Markov Chain Monte Carlo (MCMC) chains for the given model.

Usage

```
## S3 method for class 'cat_bayes'
traceplot(object, ...)
```

Arguments

`object`	A fitted model object of class `cat_bayes` that contains the Stan sampling model. The object should include the `stan_sample_model`, which is the result of fitting the model using the `rstan` package.
`...`	Additional arguments passed to the `rstan::traceplot` function. These can include customization options for the traceplot, such as `pars` for selecting specific parameters or `inc_warmup` for including or excluding warmup iterations.

Details

The function calls `rstan::traceplot` on the `stan_sample_model` contained within the `x` object. The resulting plot displays the trace of each parameter across MCMC iterations, helping to assess the convergence and mixing of the chains.

Value

A traceplot displaying the MCMC chains' trace for each parameter, helping to assess convergence.

traceplot.cat_gibbs *Traceplot for Gibbs Sampling Model*

Description

This function generates a traceplot for the Gibbs sampling model, which is typically used for posterior sampling in a Bayesian context. The traceplot visualizes the evolution of parameter values across Gibbs sampling iterations. It helps to diagnose the convergence and mixing of the chains.

Usage

```
## S3 method for class 'cat_gibbs'
traceplot(object, pars = NULL, inc_warmup = FALSE, ...)
```

Arguments

object	A fitted model object of class cat_gibbs that contains Gibbs sampling results. The object must include gibbs_iteration_log, which holds the iteration logs for all sampled parameters, and warmup and iter which indicate the warmup and total iteration counts, respectively.
pars	A character vector specifying the parameter names to plot. If NULL, the function will select the first 9 parameters automatically.
inc_warmup	A logical value indicating whether to include warmup iterations in the traceplot. If TRUE, warmup iterations are included, otherwise they are excluded. Defaults to FALSE.
...	Additional parameters to pass to other functions.

Details

The function generates a series of line plots for the selected parameters, displaying their values over the iterations of the Gibbs sampling process. If inc_warmup is set to TRUE, the traceplot includes the warmup period, otherwise, it starts after the warmup. The traceplots are arranged in a 3x3 grid, and no more than 9 parameters can be selected for plotting at once.

Value

A series of traceplots for the selected parameters, showing their evolution over the Gibbs sampling iterations.

---

update_lmm_variance *Calculates the log-likelihood for linear mixed models (LMMs) by combining observed and synthetic log-likelihoods based on the variance parameters.*

Description

This function evaluates the log-likelihood of observed and synthetic data, using residual and random-effect variance terms to determine the fit of variance parameters in the mixed model context.

Usage

```
update_lmm_variance(
  residual_variance,
  random_effect_variance,
  obs_z_eigenvalues,
  syn_z_eigenvalues,
  obs_adjusted_residuals,
  syn_adjusted_residuals,
  tau
)
```

Arguments

`residual_variance`	Numeric, the variance associated with the residual errors.
`random_effect_variance`	Numeric, the variance associated with random effects.
`obs_z_eigenvalues`	Vector, eigenvalues of the observed Z matrix of data.
`syn_z_eigenvalues`	Vector, eigenvalues of the synthetic Z matrix of data.
`obs_adjusted_residuals`	Vector, adjusted residuals of observed data.
`syn_adjusted_residuals`	Vector, adjusted residuals of synthetic data.
`tau`	Numeric, weight factor for the synthetic data.

Value

The sum of observed and synthetic log-likelihoods.

`validate_cox_initialization_input` *Validate Inputs for Catalytic Cox proportional hazards model (COX) Initialization*

Description

This function performs validation checks on input parameters for initializing a catalytic Cox proportional hazards model. It ensures that essential parameters meet requirements, such as being of the correct type, appropriate length, and having valid values.

Usage

```
validate_cox_initialization_input(
  formula,
  data,
  syn_size,
  hazard_constant,
  entry_points,
  x_degree
)
```

Arguments

formula	An object of class formula. The model formula specifying the Cox model structure. It must contain a Surv object to indicate survival analysis.
data	A data.frame containing the dataset to be used for model fitting. It should include all variables referenced in formula.
syn_size	A positive integer indicating the size of the synthetic dataset. It is recommended that this value is at least four times the number of columns in data.
hazard_constant	A positive numeric value representing the hazard constant for the Cox model.
entry_points	A numeric vector representing entry times for observations. This vector should be non-negative and have a length equal to the number of rows in data.
x_degree	A numeric vector indicating degrees for each covariate. It should be non-negative and match the number of covariates (i.e., ncol(data) - 2).

Details

This function checks:

- That syn_size, hazard_constant, entry_points, and x_degree are positive values.
- That formula includes a Surv object to be suitable for Cox models.
- That data is a data.frame.
- The complexity of formula to ensure it has fewer terms than the number of columns in data.
- The length of x_degree and entry_points to match the dimensions of data. If the conditions are not met, descriptive error messages are returned.

Value

Returns nothing if all checks pass; otherwise, raises an error.

---

validate_cox_input *Validate Inputs for Catalytic Cox Model*

Description

This function validates the parameters provided for setting up a catalytic Cox proportional hazards model with an initialization object created by cat_cox_initialization.

Usage

```
validate_cox_input(
  formula,
  cat_init,
  tau = NULL,
  tau_seq = NULL,
  init_coefficients = NULL,
```

```
  tol = NULL,
  max_iter = NULL,
  cross_validation_fold_num = NULL,
  hazard_beta = NULL,
  tau_alpha = NULL,
  tau_gamma = NULL
)
```

Arguments

formula	An object of class formula. Specifies the model structure for the Cox model, including a Surv object for survival analysis. Should at least include response variance.
cat_init	An initialization object generated by cat_cox_initialization. This object should contain necessary information about the dataset, including the time and status column names.
tau	Optional. A numeric scalar, the regularization parameter for the Cox model. Must be positive.
tau_seq	Optional. A numeric vector for specifying a sequence of regularization parameters. Must be positive.
init_coefficients	Optional. A numeric vector of initial coefficients for the Cox model. Should match the number of predictors in the dataset.
tol	Optional. A positive numeric value indicating the tolerance level for convergence in iterative fitting.
max_iter	Optional. A positive integer indicating the maximum number of iterations allowed in the model fitting.
cross_validation_fold_num	Optional. A positive integer specifying the number of folds for cross-validation. Should be greater than 1 and less than or equal to the size of the dataset.
hazard_beta	Optional. A positive numeric value representing a constant for adjusting the hazard rate in the Cox model.
tau_alpha	Optional. A positive numeric value controlling the influence of tau.
tau_gamma	Optional. A positive numeric value controlling the influence of tau_seq.

Details

This function checks:

- That tau, tol, max_iter, cross_validation_fold_num, hazard_beta, tau_alpha, and tau_gamma are positive.
- That tau_seq is a non-negative vector.
- That cat_init is generated from cat_cox_initialization.
- That formula uses the same time and status column names as those in cat_init.
- That init_coefficients has the correct length for the number of predictors.

- That `cross_validation_fold_num` is between 2 and the dataset size.
- That the dataset is sufficiently large for cross-validation, recommending fewer folds if it is not. If any conditions are not met, the function will raise an error or warning.

Value

Returns nothing if all checks pass; otherwise, raises an error or warning.

`validate_glm_initialization_input` *Validate Inputs for Catalytic Generalized Linear Models (GLMs) Initialization*

Description

This function validates the input parameters required for initializing a catalytic Generalized Linear Model (GLM). It ensures the appropriate structure and compatibility of the formula, family, data, and additional parameters before proceeding with further modeling.

Usage

```
validate_glm_initialization_input(
  formula,
  family,
  data,
  syn_size,
  custom_variance,
  gaussian_known_variance,
  x_degree
)
```

Arguments

`formula`	A formula object specifying the `stats::glm` model to be fitted. It must not contain random effects or survival terms.
`family`	A character or family object specifying the error distribution and link function. Valid values are "binomial" and "gaussian".
`data`	A `data.frame` containing the data to be used in the GLM.
`syn_size`	A positive integer specifying the sample size used for the synthetic data.
`custom_variance`	A positive numeric value for the custom variance used in the model (only applicable for Gaussian family).
`gaussian_known_variance`	A logical indicating whether the variance is known for the Gaussian family.
`x_degree`	A numeric vector specifying the degree of the predictors. Its length should match the number of predictors (excluding the response variable).

Details

This function performs the following checks:

- Ensures that `syn_size`, `custom_variance`, and `x_degree` are positive values.
- Verifies that the provided `formula` is suitable for GLMs, ensuring no random effects or survival terms.
- Checks that the provided `data` is a `data.frame`.
- Confirms that the `formula` does not contain too many terms relative to the number of columns in `data`.
- Ensures that the `family` is either "binomial" or "gaussian".
- Validates that `x_degree` has the correct length relative to the number of predictors in `data`.
- Warns if `syn_size` is too small relative to the number of columns in `data`.
- Issues warnings if `custom_variance` or `gaussian_known_variance` are used with incompatible families. If any of these conditions are not met, the function raises an error or warning to guide the user.

Value

Returns nothing if all checks pass; otherwise, raises an error or warning.

`validate_glm_input` *Validate Inputs for Catalytic Generalized Linear Models (GLMs)*

Description

This function validates the input parameters for initializing a catalytic Generalized Linear Models (GLMs). It ensures that the provided model formula, family, and additional parameters are suitable for further analysis. The function performs various checks on the input values to confirm they meet expected criteria.

Usage

```
validate_glm_input(
  formula,
  cat_init,
  tau = NULL,
  tau_seq = NULL,
  tau_0 = NULL,
  parametric_bootstrap_iteration_times = NULL,
  cross_validation_fold_num = NULL,
  risk_estimate_method = NULL,
  discrepancy_method = NULL,
  binomial_joint_theta = FALSE,
  binomial_joint_alpha = FALSE,
  binomial_tau_lower = NULL,
```

```
  tau_alpha = NULL,
  tau_gamma = NULL,
  gibbs_iter = NULL,
  gibbs_warmup = NULL,
  coefs_iter = NULL,
  gaussian_variance_alpha = NULL,
  gaussian_variance_beta = NULL
)
```

Arguments

formula	A formula object specifying the GLM to be fitted. The left-hand side of the formula should at least contains the response variable.
cat_init	An object of class cat_initialization generated by cat_glm_initialization. It contains model initialization details, such as the response variable name and the GLM family.
tau	A positive numeric value for the tau parameter in the model. It represents a regularization or scaling factor and must be greater than zero.
tau_seq	A numeric vector specifying a sequence of tau values. This is used for parameter tuning and must contain positive values.
tau_0	A positive numeric value for the initial tau parameter, which must be greater than zero.
parametric_bootstrap_iteration_times	An integer specifying the number of iterations for the parametric bootstrap method. It must be greater than zero.
cross_validation_fold_num	An integer for the number of folds in cross-validation. It must be greater than 1 and less than or equal to the number of observations.
risk_estimate_method	A character string specifying the method for estimating risk, such as "parametric_bootstrap" or other options, depending on the family of the GLM.
discrepancy_method	A character string specifying the method for calculating discrepancy. The valid options depend on the GLM family and risk estimation method.
binomial_joint_theta	Logical; if TRUE, uses joint theta (theta = 1/tau) in Binomial models.
binomial_joint_alpha	Logical; if TRUE, uses joint alpha (adaptive tau_alpha) in Binomial models.
binomial_tau_lower	A positive numeric value specifying the lower bound for tau in binomial GLMs. It must be greater than zero.
tau_alpha	A positive numeric value for the tau alpha parameter.
tau_gamma	A positive numeric value for the tau gamma parameter.
gibbs_iter	An integer for the number of Gibbs iterations in the sampling process. It must be greater than zero.

`gibbs_warmup` An integer for the number of warm-up iterations in the Gibbs sampling. It must be positive and less than the total number of iterations.

`coefs_iter` An integer specifying the number of iterations for the coefficient update in the Gibbs sampling. It must be positive.

`gaussian_variance_alpha`
The shape parameter for the inverse-gamma prior on variance if the variance is unknown in Gaussian models. It must be positive.

`gaussian_variance_beta`
The scale parameter for the inverse-gamma prior on variance if the variance is unknown in Gaussian models. It must be positive.

Details

This function performs several checks to ensure the validity of the input parameters:

- Ensures that `tau`, `tau_0`, `parametric_bootstrap_iteration_times`, `binomial_tau_lower`, `tau_alpha`, `tau_gamma`, `gibbs_iter`, `gibbs_warmup`, and `coefs_iter` are positive values.
- Verifies that `cat_init` is an object generated by `cat_glm_initialization`.
- Checks that the `formula` response name matches the response name used in the `cat_init` object.
- Verifies that `risk_estimate_method` and `discrepancy_method` are compatible with the GLM family and that no invalid combinations are used.
- Warns if the dataset size is too large for the specified risk estimation method. If any of these conditions are not met, the function raises an error or warning to guide the user.

Value

Returns nothing if all checks pass; otherwise, raises an error or warning.

---

validate_lmm_initialization_input

Validate Inputs for Catalytic Linear Mixed Model (LMM) Initialization

---

Description

This function validates the parameters needed for initializing a catalytic Linear Mixed Model (LMM) or Generalized Linear Model (GLM) based on the input formula, data, and column specifications.

Usage

```
validate_lmm_initialization_input(
  formula,
  data,
  x_cols,
  y_col,
  z_cols,
```

```
  group_col,
  syn_size
)
```

Arguments

formula	An object of class formula representing the model formula, typically including fixed and random effects for LMMs or for GLMs.
data	A data.frame containing the data for model fitting. This should include all columns specified in x_cols, y_col, z_cols, and group_col.
x_cols	A character vector of column names to be used as predictor variables in the model.
y_col	A single character string specifying the name of the response variable column.
z_cols	A character vector of column names to be used as additional predictors or grouping factors, depending on the model structure.
group_col	A single character string specifying the name of the grouping variable for random effects.
syn_size	Optional. A positive integer indicating the synthetic data size, typically for use in data augmentation or model diagnostics.

Details

This function performs the following checks:

- Ensures syn_size is a positive integer.
- Verifies that formula is not for survival analysis (e.g., does not contain Surv terms).
- Checks that the formula is not overly complex by confirming it has fewer terms than the total columns in data.
- Ensures y_col and group_col each contain only one column name.
- Confirms data is a data.frame.
- Validates that all specified columns in x_cols, y_col, z_cols, and group_col exist in data without overlap or missing values.
- Warns if syn_size is set too small relative to the data dimensions, recommending a larger value. If any of these conditions are not met, the function raises an error or warning to guide the user.

Value

Returns nothing if all checks pass; otherwise, raises an error or warning.

validate_lmm_input *Validate Inputs for Catalytic Linear Mixed Model (LMM)*

Description

This function validates the parameters needed for fitting a catalytic Linear Mixed Model (LMM) or Generalized Linear Model (GLM), specifically for the use with the categorical initialization from `cat_lmm_initialization`.

Usage

```
validate_lmm_input(
  cat_init,
  tau = NULL,
  residual_variance_0 = NULL,
  random_effect_variance_0 = NULL,
  coefs_0 = NULL,
  optimize_domain = NULL,
  max_iter = NULL,
  tol = NULL,
  tau_seq = NULL,
  cross_validation_fold_num = NULL
)
```

Arguments

`cat_init`	An object of class `cat_initialization`, typically generated from the `cat_lmm_initialization` function.
`tau`	A positive numeric value specifying the penalty parameter for the model.
`residual_variance_0`	A positive numeric value for the initial residual variance estimate.
`random_effect_variance_0`	A positive numeric value for the initial random effect variance estimate.
`coefs_0`	A numeric vector of length equal to the number of columns in the observation matrix. This represents the initial values for the model coefficients.
`optimize_domain`	A numeric vector of length 2 specifying the domain for the optimization procedure.
`max_iter`	A positive integer specifying the maximum number of iterations for the optimization.
`tol`	A positive numeric value indicating the tolerance level for convergence.
`tau_seq`	A numeric vector representing a sequence of values for the penalty parameter.
`cross_validation_fold_num`	A positive integer specifying the number of folds for cross-validation.

Details

This function performs the following checks:

- Ensures that `tau`, `tau_seq`, `residual_variance_0`, `random_effect_variance_0`, `optimize_domain`, `max_iter`, and `tol` are positive values.
- Verifies that `cat_init` is an object generated by `cat_lmm_initialization`.
- Checks if `coefs_0` has the same length as the number of columns in the observation matrix of `cat_init`.
- Ensures `optimize_domain` is a numeric vector of length 2.
- Confirms that `cross_validation_fold_num` is greater than 1 and less than the number of observations in `cat_init`. If any of these conditions are not met, the function raises an error to guide the user.

Value

Returns nothing if all checks pass; otherwise, raises an error.

---

validate_positive *Validate Positive or Non-negative Parameter*

Description

This function checks whether a parameter value is positive (or non-negative if `incl_0` is set to `TRUE`). It can handle both single numeric values and vectors, and it raises an error with an informative message if the validation fails.

Usage

```
validate_positive(param_name, param_value, incl_0 = FALSE, is_vector = FALSE)
```

Arguments

`param_name`	A string representing the name of the parameter. Used in the error message.
`param_value`	The parameter value to validate, either a single numeric or a numeric vector.
`incl_0`	Logical, if `TRUE`, allows non-negative values (larger or equal to 0); if `FALSE`, requires positive values (larger than 0).
`is_vector`	Logical, if `TRUE`, treats `param_value` as a vector; otherwise, expects a single numeric value.

Value

`NULL` if validation passes; otherwise, an error is raised.

Index