Package 'ARTtransfer'

May 6, 2026

Type Package

Title Adaptive and Robust Pipeline for Transfer Learning

Version 1.0.0

Date 2024-10-16

Description Adaptive and Robust Transfer Learning (ART) is a flexible framework for transfer learning that integrates information from auxiliary data sources to improve model performance on primary tasks. It is designed to be robust against negative transfer by including the non-transfer model in the candidate pool, ensuring stable performance even when auxiliary datasets are less informative. See the paper, Wang, Wu, and Ye (2023) <doi:10.1002/sta4.582>.

Imports gbm, glmnet, nnet, randomForest, stats

Encoding UTF-8

License GPL-2

VignetteBuilder knitr

Suggests knitr, rmarkdown

Repository CRAN

RoxygenNote 7.3.0

NeedsCompilation no

Author Boxiang Wang [aut, cre],
Yunan Wu [aut],
Chenglong Ye [aut]

Maintainer Boxiang Wang <boxiang-wang@uiowa.edu>

Date/Publication 2024-10-24 15:10:14 UTC

Contents

ART *ART: Adaptive and Robust Transfer Learning*

Description

ART is a flexible framework for transfer learning that leverages information from auxiliary data sources to enhance model performance on primary tasks. It is designed to be robust against negative transfer by including the non-transfer model in the candidate pool, ensuring stable performance even when auxiliary datasets provide limited or no useful information. The ART framework supports both regression and classification tasks, aggregating predictions from multiple auxiliary models and the primary model using an adaptive exponential weighting mechanism to prevent negative transfer. Variable importance is also provided to indicate the contribution of each variable in the final model.

Usage

```
ART(
  X,
  y,
  X_aux,
  y_aux,
  X_test,
  func,
  lam = 1,
  maxit = 5000L,
  eps = 1e-06,
  type = c("regression", "classification"),
  is_coef = TRUE,
  importance = TRUE,
  ...
)
```

Arguments

X	A matrix for the primary dataset (target domain) predictors.
y	A vector for the primary dataset (target domain) responses.
X_aux	A list of matrices for the auxiliary datasets (source domains) predictors.
y_aux	A list of vectors for the auxiliary datasets (source domains) responses.

X_test	A matrix for the test dataset predictors.
func	A function used to fit the model on each dataset. The function must have the following signature: `func(X, y, X_val, y_val, X_test, min_prod = 1e-5, max_prod = 1-1e-5, ...)`. The function should return a list with the following elements: • dev: The deviance (or loss) on the validation set if provided. • pred: The predictions on the test set if X_test is provided. • coef (optional): The model coefficients (only for regression models when is_coef = TRUE). Pre-built wrapper functions, such as fit_lm, fit_logit, fit_glmnet_lm, fit_glmnet_logit, fit_random_forest, fit_gbm, and fit_nnet, can be used. Users may also provide their own model-fitting functions, but the input and output structure must follow the described signature and format.
lam	A regularization parameter for weighting the auxiliary models. Default is 1.
maxit	The maximum number of iterations for the model. Default is 5000.
eps	A convergence threshold for stopping the iterations. Default is 1e-6.
type	A string specifying the task type. Options are "regression" or "classification". Default is "regression".
is_coef	Logical; if TRUE, coefficients from the model are returned. Default is TRUE.
importance	Logical; if TRUE, variable importance is calculated. Only applicable if 'is_coef' is TRUE. Default is TRUE.
...	Additional arguments passed to the model-fitting function.

Details

The ART function performs adaptive and robust transfer learning by iteratively combining predictions from the primary dataset and auxiliary datasets. It updates the weights of each dataset's predictions through an aggregation process, eventually yielding a final set of predictions based on weighted contributions from the source and target models.

The auxiliary datasets ('X_aux' and 'y_aux') must be provided as lists, with each element corresponding to a dataset from a different source domain.

Value

A list containing:

pred_ART	The predictions for the test dataset.
coef_ART	The coefficients of the final model, if 'is_coef' is TRUE.
W_ART	The final weights for each dataset (including the primary dataset).
iter_ART	The number of iterations performed until convergence.
VI_ART	The variable importance, if 'importance' is TRUE.

Examples

```
# Example usage
dat <- generate_data(n0=50, K=3, nk=50, K_noise=2, nk_noise=30, p=10,
       mu_trgt=1, xi_aux=0.5, ro=0.5, err_sig=1)
fit <- ART(dat$X, dat$y, dat$X_aux, dat$y_aux, dat$X_test, func=fit_lm, lam=1, type="regression")
```

ART_I_AM *ART_I_AM: ART-Integrated-Aggregrating Machines*

Description

'ART_I_AM' performs adaptive and robust transfer learning through the aggregration of multiple machine learning models, specifically random forests, AdaBoost, and neural networks. This method aggregates the predictions from these models across multiple auxiliary datasets and the primary dataset to enhance model performance on the primary task. Users do not need to specify the models in the function, while the framework is general and users can write their own function integrating other machine learning models.

Usage

```
ART_I_AM(X, y, X_aux, y_aux, X_test, lam = 1, maxit = 5000L, eps = 1e-06, ...)
```

Arguments

X	A matrix for the primary dataset (target domain) predictors.
y	A vector for the primary dataset (target domain) responses.
X_aux	A list of matrices for the auxiliary datasets (source domains) predictors.
y_aux	A list of vectors for the auxiliary datasets (source domains) responses.
X_test	A matrix for the test dataset predictors.
lam	A regularization parameter for weighting the auxiliary models. Default is 1.
maxit	The maximum number of iterations for the aggregation process. Default is 5000.
eps	A convergence threshold for stopping the iterations. Default is 1e-6.
...	Not used in ART_I_AM.

Details

The 'ART_I_AM' function automatically integrates three machine learning models: - Random Forest ('fit_rf') - AdaBoost ('fit_gbm') - Neural Network ('fit_nnet')

These models are applied to both the primary dataset and auxiliary datasets. The function aggregates the predictions of each model using adaptive weights determined by the exponential weighting scheme. The aggregation improves the prediction power by considering different models and data simultaneously.

Value

A list containing:

pred_ART	The predictions for the test dataset aggregated from the different models and datasets.
W_ART	The final weights for each model and dataset combination.
iter_ART	The number of iterations performed until convergence.

---

fit_gbm *fit_gbm: Gradient Boosting Wrapper for the ARTtransfer package*

Description

This function fits a gradient boosting model using ‘gbm()‘ from the R package gbm. It returns the deviance on a validation set and predictions on a test set. It is designed for use in the ‘ART‘ adaptive and robust transfer learning framework.

Usage

```
fit_gbm(
  X,
  y,
  X_val,
  y_val,
  X_test,
  min_prod = 1e-05,
  max_prod = 1 - 1e-05,
  ...
)
```

Arguments

X	A matrix of predictors for the training set.
y	A vector of binary responses for the training set.
X_val	A matrix of predictors for the validation set. If ‘NULL‘, deviance is not calculated.
y_val	A vector of binary responses for the validation set. If ‘NULL‘, deviance is not calculated.
X_test	A matrix of predictors for the test set. If ‘NULL‘, predictions are not generated.
min_prod	A numeric value indicating the minimum probability bound for predictions. Default is ‘1e-5‘.
max_prod	A numeric value indicating the maximum probability bound for predictions. Default is ‘1-1e-5‘.
...	Additional arguments passed to the ‘gbm()‘ function.

Value

A list containing:

dev	The deviance (negative log-likelihood) on the validation set if provided, otherwise 'NULL'.
pred	The predicted probabilities on the test set if 'X_test' is provided, otherwise 'NULL'.

Examples

```
# Fit a gradient boosting model with validation and test data
X_train <- matrix(rnorm(100 * 5), 100, 5)
y_train <- rbinom(100, 1, 0.5)
X_val <- matrix(rnorm(50 * 5), 50, 5)
y_val <- rbinom(50, 1, 0.5)
X_test <- matrix(rnorm(20 * 5), 20, 5)

fit <- fit_gbm(X_train, y_train, X_val, y_val, X_test)
```

---

fit_glmnet_lm *fit_glmnet_lm: Sparse Linear Regression Wrapper for the ARTtransfer package*

Description

This function fits a sparse linear regression model using 'glmnet()' from the R package glmnet for regression. It returns the coefficients, deviance on a validation set, and predictions on a test set. It is designed for use in the 'ART' adaptive and robust transfer learning framework.

Usage

```
fit_glmnet_lm(
  X,
  y,
  X_val,
  y_val,
  X_test,
  min_prod = 1e-05,
  max_prod = 1 - 1e-05,
  nfolds = 5,
  ...
)
```

Arguments

X	A matrix of predictors for the training set.
y	A vector of responses for the training set.
X_val	A matrix of predictors for the validation set. If 'NULL', deviance is not calculated.
y_val	A vector of responses for the validation set. If 'NULL', deviance is not calculated.
X_test	A matrix of predictors for the test set. If 'NULL', predictions are not generated.
min_prod	A numeric value indicating the minimum probability bound for predictions (not used in this function but passed for compatibility). Default is '1e-5'.
max_prod	A numeric value indicating the maximum probability bound for predictions (not used in this function but passed for compatibility). Default is '1-1e-5'.
nfolds	An integer specifying the number of folds for cross-validation. Default is 5.
...	Additional arguments passed to the function.

Value

A list containing:

dev	The mean squared error (deviance) on the validation set if provided, otherwise 'NULL'.
pred	The predictions on the test set if 'X_test' is provided, otherwise 'NULL'.
coef	The fitted coefficients of the sparse linear model.

Examples

```
# Fit a sparse linear model with validation and test data
X_train <- matrix(rnorm(100 * 5), 100, 5)
y_train <- X_train %*% rnorm(5) + rnorm(100)
X_val <- matrix(rnorm(50 * 5), 50, 5)
y_val <- X_val %*% rnorm(5) + rnorm(50)
X_test <- matrix(rnorm(20 * 5), 20, 5)

fit <- fit_glmnet_lm(X_train, y_train, X_val, y_val, X_test)
```

---

fit_glmnet_logit *fit_glmnet_logit: Sparse Logistic Regression Wrapper for the ART-transfer package*

Description

This function fits a sparse logistic regression model using 'glmnet()' from the R package glmnet for classification. It returns the coefficients, deviance on a validation set, and predictions on a test set. It is designed for use in the 'ART' adaptive and robust transfer learning framework.

Usage

```
fit_glmnet_logit(
  X,
  y,
  X_val,
  y_val,
  X_test,
  min_prod = 1e-05,
  max_prod = 1 - 1e-05,
  nfolds = 5,
  ...
)
```

Arguments

X	A matrix of predictors for the training set.
y	A vector of binary responses for the training set.
X_val	A matrix of predictors for the validation set. If 'NULL', deviance is not calculated.
y_val	A vector of binary responses for the validation set. If 'NULL', deviance is not calculated.
X_test	A matrix of predictors for the test set. If 'NULL', predictions are not generated.
min_prod	A numeric value indicating the minimum probability bound for predictions. Default is '1e-5'.
max_prod	A numeric value indicating the maximum probability bound for predictions. Default is '1-1e-5'.
nfolds	An integer specifying the number of folds for cross-validation. Default is 5.
...	Additional arguments passed to the function.

Value

A list containing:

dev	The deviance (negative log-likelihood) on the validation set if provided, otherwise 'NULL'.
pred	The predicted probabilities on the test set if 'X_test' is provided, otherwise 'NULL'.
coef	The fitted coefficients of the sparse logistic model.

Examples

```
# Fit a sparse logistic regression model with validation and test data
X_train <- matrix(rnorm(100 * 5), 100, 5)
y_train <- rbinom(100, 1, 0.5)
X_val <- matrix(rnorm(50 * 5), 50, 5)
y_val <- rbinom(50, 1, 0.5)
```

```
X_test <- matrix(rnorm(20 * 5), 20, 5)

fit <- fit_glmnet_logit(X_train, y_train, X_val, y_val, X_test)
```

`fit_lm` *fit_lm: Linear Regression Wrapper for the ARTtransfer package*

Description

This function fits a linear regression model using ‘lm()‘ and returns the coefficients, deviance on a validation set, and predictions on a test set. It is specifically designed for use in the ‘ART‘ adaptive and robust transfer learning framework.

Usage

```
fit_lm(X, y, X_val, y_val, X_test, min_prod = 1e-05, max_prod = 1 - 1e-05, ...)
```

Arguments

`X`	A matrix of predictors for the training set.
`y`	A vector of responses for the training set.
`X_val`	A matrix of predictors for the validation set. If ‘NULL‘, deviance is not calculated.
`y_val`	A vector of responses for the validation set. If ‘NULL‘, deviance is not calculated.
`X_test`	A matrix of predictors for the test set. If ‘NULL‘, predictions are not generated.
`min_prod`	A numeric value indicating the minimum probability bound for predictions (not used in this function but passed for compatibility). Default is ‘1e-5‘.
`max_prod`	A numeric value indicating the maximum probability bound for predictions (not used in this function but passed for compatibility). Default is ‘1-1e-5‘.
`...`	Additional arguments passed to the function (currently unused).

Value

A list containing:

`dev`	The mean squared error (deviance) on the validation set if provided, otherwise ‘NULL‘.
`pred`	The predictions on the test set if ‘X_test‘ is provided, otherwise ‘NULL‘.
`coef`	The fitted coefficients of the linear model.

Examples

```
# Fit a linear model with validation and test data
X_train <- matrix(rnorm(100 * 5), 100, 5)
y_train <- X_train %*% rnorm(5) + rnorm(100)
X_val <- matrix(rnorm(50 * 5), 50, 5)
y_val <- X_val %*% rnorm(5) + rnorm(50)
X_test <- matrix(rnorm(20 * 5), 20, 5)

fit <- fit_lm(X_train, y_train, X_val, y_val, X_test)
```

`fit_logit` *fit_logit: Logistic Regression Wrapper for the ARTtransfer package*

Description

This function fits a logistic regression model using ‘glm()‘ and returns the coefficients, deviance on a validation set, and predictions on a test set. It is specifically designed for use in the ‘ART‘ adaptive and robust transfer learning framework.

Usage

```
fit_logit(
  X,
  y,
  X_val,
  y_val,
  X_test,
  min_prod = 1e-05,
  max_prod = 1 - 1e-05,
  ...
)
```

Arguments

`X`	A matrix of predictors for the training set.
`y`	A vector of binary responses for the training set.
`X_val`	A matrix of predictors for the validation set. If ‘NULL‘, deviance is not calculated.
`y_val`	A vector of binary responses for the validation set. If ‘NULL‘, deviance is not calculated.
`X_test`	A matrix of predictors for the test set. If ‘NULL‘, predictions are not generated.
`min_prod`	A numeric value indicating the minimum probability bound for predictions. Default is ‘1e-5‘.
`max_prod`	A numeric value indicating the maximum probability bound for predictions. Default is ‘1-1e-5‘.
`...`	Additional arguments passed to the function (currently unused).

Value

A list containing:

dev	The deviance (negative log-likelihood) on the validation set if provided, otherwise 'NULL'.
pred	The predicted probabilities on the test set if 'X_test' is provided, otherwise 'NULL'.
coef	The fitted coefficients of the logistic model.

Examples

```
# Fit a logistic regression model with validation and test data
X_train <- matrix(rnorm(100 * 5), 100, 5)
y_train <- rbinom(100, 1, 0.5)
X_val <- matrix(rnorm(50 * 5), 50, 5)
y_val <- rbinom(50, 1, 0.5)
X_test <- matrix(rnorm(20 * 5), 20, 5)

fit <- fit_logit(X_train, y_train, X_val, y_val, X_test)
```

---

fit_nnet *fit_nnet: Neural Network Wrapper for the ARTtransfer package*

Description

This function fits a neural network model using 'nnet()' from the R package nnet. It returns the deviance on a validation set and predictions on a test set. It is designed for use in the 'ART' adaptive and robust transfer learning framework.

Usage

```
fit_nnet(
  X,
  y,
  X_val,
  y_val,
  X_test,
  min_prod = 1e-05,
  max_prod = 1 - 1e-05,
  ...
)
```

Arguments

X	A matrix of predictors for the training set.
y	A vector of binary responses for the training set.
X_val	A matrix of predictors for the validation set. If 'NULL', deviance is not calculated.
y_val	A vector of binary responses for the validation set. If 'NULL', deviance is not calculated.
X_test	A matrix of predictors for the test set. If 'NULL', predictions are not generated.
min_prod	A numeric value indicating the minimum probability bound for predictions. Default is '1e-5'.
max_prod	A numeric value indicating the maximum probability bound for predictions. Default is '1-1e-5'.
...	Additional arguments passed to 'nnet()'.

Value

A list containing:

dev	The deviance (negative log-likelihood) on the validation set if provided, otherwise 'NULL'.
pred	The predicted probabilities on the test set if 'X_test' is provided, otherwise 'NULL'.

Examples

```
# Fit a neural network model with validation and test data
X_train <- matrix(rnorm(100 * 5), 100, 5)
y_train <- rbinom(100, 1, 0.5)
X_val <- matrix(rnorm(50 * 5), 50, 5)
y_val <- rbinom(50, 1, 0.5)
X_test <- matrix(rnorm(20 * 5), 20, 5)

fit <- fit_nnet(X_train, y_train, X_val, y_val, X_test)
```

---

fit_rf *fit_rf: Random Forest Wrapper for the ARTtransfer package*

Description

This function fits a random forest classification model using 'randomForest()' from the R package randomForest. It returns the deviance on a validation set and predictions on a test set. It is designed for use in the 'ART' adaptive and robust transfer learning framework.

Usage

```
fit_rf(X, y, X_val, y_val, X_test, min_prod = 1e-05, max_prod = 1 - 1e-05, ...)
```

Arguments

X	A matrix of predictors for the training set.
y	A vector of binary responses for the training set.
X_val	A matrix of predictors for the validation set. If 'NULL', deviance is not calculated.
y_val	A vector of binary responses for the validation set. If 'NULL', deviance is not calculated.
X_test	A matrix of predictors for the test set. If 'NULL', predictions are not generated.
min_prod	A numeric value indicating the minimum probability bound for predictions. Default is '1e-5'.
max_prod	A numeric value indicating the maximum probability bound for predictions. Default is '1-1e-5'.
...	Additional arguments passed to the 'randomForest()' function.

Value

A list containing:

dev	The deviance (negative log-likelihood) on the validation set if provided, otherwise 'NULL'.
pred	The predicted probabilities on the test set if 'X_test' is provided, otherwise 'NULL'.

Examples

```
# Fit a random forest model with validation and test data
X_train <- matrix(rnorm(100 * 5), 100, 5)
y_train <- rbinom(100, 1, 0.5)
X_val <- matrix(rnorm(50 * 5), 50, 5)
y_val <- rbinom(50, 1, 0.5)
X_test <- matrix(rnorm(20 * 5), 20, 5)

fit <- fit_rf(X_train, y_train, X_val, y_val, X_test)
```

generate_data *generate_data: Generate synthetic primary, auxiliary, and noisy datasets for transfer learning*

Description

This function generates synthetic datasets for the primary task (target domain), auxiliary datasets (source domains), and noisy datasets for use in transfer learning simulations. It allows flexible input for the sizes of the auxiliary and noisy datasets, supports different covariance structures, and can optionally generate test datasets. Users can specify true coefficients or rely on random generation. The function supports generating datasets for both regression and binary classification tasks.

Usage

```
generate_data(
  n0,
  p,
  K,
  nk,
  is_noise = TRUE,
  K_noise = 2,
  nk_noise = 30,
  mu_trgt,
  xi_aux,
  ro,
  err_sig,
  true_beta = NULL,
  noise_beta = NULL,
  Sigma_type = "AR",
  is_test = TRUE,
  n_test = n0,
  task = "regression"
)
```

Arguments

n0	An integer specifying the number of observations in the primary dataset (target domain).
p	An integer specifying the dimension, namely the number of predictors. All the generated data must have the same dimension.
K	An integer specifying the number of auxiliary datasets (source domains).
nk	Either an integer specifying the number of observations in each auxiliary dataset (source domains), or a vector where each element specifies the size of the corresponding auxiliary dataset. If ‘nk‘ is a vector, its length must match the number of auxiliary datasets (‘K‘).

is_noise	Logical; if TRUE, includes noisy data. If FALSE, ‘K_noise‘ and ‘nk_noise‘ are ignored. Default is TRUE.
K_noise	An integer specifying the number of noisy auxiliary datasets. If ‘K_noise = 0‘, noisy datasets are skipped. If ‘is_noise = FALSE‘, this argument is not used.
nk_noise	Either an integer specifying the number of observations in each noisy dataset, or a vector where each element specifies the size of the corresponding noisy dataset. If ‘nk_noise‘ is a vector, its length must match the number of noisy datasets (‘K_noise‘).
mu_trgt	A numeric value specifying the mean of the true coefficients in the primary dataset.
xi_aux	A numeric value representing the shift applied to the true coefficients in the auxiliary datasets.
ro	A numeric value representing the correlation between predictors (applies to the covariance matrix).
err_sig	A numeric value specifying the standard deviation of the noise added to the response.
true_beta	A vector of true coefficients for the primary dataset. If ‘NULL‘, it is randomly generated. Default is ‘NULL‘.
noise_beta	A vector of noise coefficients. If ‘NULL‘, it is set to ‘-true_beta‘. Default is ‘NULL‘.
Sigma_type	A string specifying the covariance structure for the predictors. Options are: "AR" (auto-regressive structure) or "CS" (compound symmetry structure). Default is "AR".
is_test	Logical; if TRUE, generates test dataset (‘X_test‘, ‘y_test‘). Default is TRUE.
n_test	An integer specifying the number of observations in the test data. Default is n0.
task	A string specifying the type of task. Options are "regression" or "classification". Default is "regression".

Details

The function first generates a covariance matrix based on the specified ‘Sigma_type‘, then creates the primary dataset (‘X‘, ‘y‘), the auxiliary datasets (‘X_aux‘, ‘y_aux‘), and optionally generates test datasets (‘X_test‘, ‘y_test‘). The auxiliary datasets are combined with noisy datasets into ‘X_aux‘ and ‘y_aux‘ for transfer learning use.

If ‘is_noise = FALSE‘, then no noisy data is generated and ‘K_noise‘ and ‘nk_noise‘ are ignored. If ‘K_noise = 0‘, noisy data is skipped regardless of the value of ‘is_noise‘. The task can be either "regression" or "classification". In classification mode, binary response variables are generated using a logistic function.

If ‘nk‘ or ‘nk_noise‘ is a vector, it checks if its length matches the number of auxiliary or noisy +, respectively. If the lengths do not match, an error is returned.

Value

A list containing:

X	The primary dataset predictors (target domain).
y	The primary dataset responses (target domain).
X_aux	A list of matrices combining auxiliary and noisy dataset predictors.
y_aux	A list of vectors combining auxiliary and noisy dataset responses.
X_test	The test dataset predictors, if 'is_test=TRUE'.
y_test	The test dataset responses, if 'is_test=TRUE'.

Examples

```
# Example: Generate data with auxiliary, noisy, and test datasets for regression
dat_reg <- generate_data(n0=100, p=10, K=3, nk=50, is_noise=TRUE, K_noise=2, nk_noise=30,
                         mu_trgt=1, xi_aux=0.5, ro=0.3, err_sig=1,
                         is_test=TRUE, task="regression")

# Example: Generate data with auxiliary, noisy, and test datasets for classification
dat_class <- generate_data(n0=100, p=10, K=3, nk=50, is_noise=TRUE, K_noise=2, nk_noise=30,
                           mu_trgt=1, xi_aux=0.5, ro=0.3, err_sig=1,
                           is_test=TRUE, task="classification")

# Display the dimensions of the generated data
cat("Primary dataset (X):", dim(dat_reg$X), "\n")   # Should print 100 x 10 for regression
cat("Primary dataset (y):", length(dat_reg$y), "\n") # Should print length 100 for regression

# Display the dimensions of auxiliary datasets
cat("Auxiliary dataset 1 (X_aux[[1]]):", dim(dat_reg$X_aux[[1]]), "\n") # Should print 50 x 10
cat("Auxiliary dataset 2 (X_aux[[2]]):", dim(dat_reg$X_aux[[2]]), "\n") # Should print 50 x 10

# Display the dimensions of noisy datasets (if generated)
cat("Noisy dataset 1 (X_aux[[4]]):", dim(dat_reg$X_aux[[4]]), "\n") # Should print 30 x 10

# Display test data dimensions (if generated)
if (!is.null(dat_reg$X_test)) {
  cat("Test dataset (X_test):", dim(dat_reg$X_test), "\n") # Should print 100 x 10
  cat("Test dataset (y_test):", length(dat_reg$y_test), "\n") # Should print length 100
}
```

---

stan *stan: Standardize Training, Validation, and Test Datasets*

Description

This function standardizes the training, validation, and test datasets by centering and scaling them using the mean and standard deviation from the training set. It ensures that the validation and test sets are transformed using the same parameters derived from the training data.

Usage

```
stan(train, validation = NULL, test = NULL)
```

Arguments

train	A list containing the training set. The list must have a component 'X' for predictors.
validation	A list containing the validation set. The list must have a component 'X' for predictors. If 'NULL', the validation set is not standardized. Default is 'NULL'.
test	A list containing the test set. The list must have a component 'X' for predictors. If 'NULL', the test set is not standardized. Default is 'NULL'.

Value

A list with the following components:

train	The standardized training set, with predictors centered and scaled.
validation	The standardized validation set (if provided), standardized using the training set's mean and standard deviation.
test	The standardized test set (if provided), standardized using the training set's mean and standard deviation.

Examples

```
# Example usage
train_data <- list(X = matrix(rnorm(100), ncol=10))
validation_data <- list(X = matrix(rnorm(50), ncol=10))
test_data <- list(X = matrix(rnorm(50), ncol=10))

standardized <- stan(train = train_data, validation = validation_data, test = test_data)
```

Index