Package 'SCE'

June 23, 2025

Title Stepwise Clustered Ensemble

Version 1.0.0

Description Implementation of Stepwise Clustered Ensemble (SCE) and Stepwise Cluster Analysis (SCA) for multivariate data analysis. The package provides comprehensive tools for feature selection, model training, prediction, and evaluation in hydrological and environmental modeling applications. Key functionalities include recursive feature elimination (RFE), Wilks feature importance analysis, model validation through out-of-bag (OOB) validation, and ensemble prediction capabilities. The package supports both single and multivariate response variables, making it suitable for complex environmental modeling scenarios. For more details see Li et al. (2021) <doi:10.5194/hess-25-4947-2021>.

URL `https://doi.org/10.5194/hess-25-4947-2021`

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 3.5.0)

Imports stats (>= 3.5.0), utils (>= 3.5.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown

NeedsCompilation no

Author Kailong Li [aut, cre]

Maintainer Kailong Li <`lkl98509509@gmail.com`>

Repository CRAN

Date/Publication 2025-06-23 10:50:02 UTC

Contents

Air_quality_testing 2
Air_quality_training 3
Model_evaluation 4
Model_simulation 5
OOB_validation 7
RFE_SCE 8

SCA 10
SCA_importance 13
SCA_tree_predict 15
SCE 17
SCE_Prediction 20
Streamflow_testing_10var 21
Streamflow_testing_22var 22
Streamflow_training_10var 24
Streamflow_training_22var 25
Wilks_importance 26

Index **29**

`Air_quality_testing` *Example Air Quality Testing Dataset for the SCE Model*

Description

The "Air_quality_testing" dataset serves as an example dataset to illustrate the functioning and application of the SCE model within the SCE package. It contains various air quality parameters and meteorological variables measured on a monthly scale, which are used as inputs in the SCE model for model testing.

Usage

```
data("Air_quality_testing")
```

Format

A data frame with 8760 rows and 7 variables:

Date Date and time of measurement (POSIXct format)

PM2.5 Particulate matter with diameter less than 2.5 micrometers (\mu g/m^3).

PM10 Particulate matter with diameter less than 10 micrometers (\mu g/m^3).

SO2 Sulfur dioxide concentration (\mu g/m^3).

NO2 Nitrogen dioxide concentration (\mu g/m^3).

CO Carbon monoxide concentration (\mu g/m^3).

O3 Ozone concentration (\mu g/m^3).

TEMP Temperature (\textdegree C).

PRES Atmospheric pressure (hPa).

DEWP Dew point temperature (\textdegree C).

RAIN Precipitation amount (mm).

WSPM Wind speed (m/s).

Source

Air quality monitoring stations

Examples

```
data(Air_quality_testing)
head(Air_quality_testing)
```

`Air_quality_training` *Air Quality Training Dataset*

Description

This dataset contains air quality measurements for training purposes. It includes various air pollutant concentrations measured at different locations and times.

Usage

```
data("Air_quality_training")
```

Format

A data frame with 8760 rows and 7 variables:

Date Date and time of measurement (POSIXct format)

PM2.5 Particulate matter with diameter less than 2.5 micrometers (\mu g/m^3).

PM10 Particulate matter with diameter less than 10 micrometers (\mu g/m^3).

SO2 Sulfur dioxide concentration (\mu g/m^3).

NO2 Nitrogen dioxide concentration (\mu g/m^3).

CO Carbon monoxide concentration (\mu g/m^3).

O3 Ozone concentration (\mu g/m^3).

TEMP Temperature (\textdegree C).

PRES Atmospheric pressure (hPa).

DEWP Dew point temperature (\textdegree C).

RAIN Precipitation amount (mm).

WSPM Wind speed (m/s).

Source

Air quality monitoring stations

Examples

```
data(Air_quality_training)
head(Air_quality_training)
```

`Model_evaluation` *Evaluate SCE and SCA Model Performances*

Description

These functions evaluate the performance of SCE and SCA models using six distinct metrics: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Nash-Sutcliffe Efficiency (NSE), Log-transformed NSE, R-squared, and Kling-Gupta Efficiency (KGE). The functions handle both single and multiple predictants, with comprehensive input validation and proper handling of NaN values and zero/negative values.

Usage

```
SCA_Model_evaluation(Testing_data, Simulations, Predictant, digits = 3)

SCE_Model_evaluation(Testing_data, Training_data, Simulations, Predictant, digits = 3)
```

Arguments

`Testing_data`	A data.frame containing the observations used during model testing. Must include all specified predictants.
`Training_data`	A data.frame containing the observations used during model training. Required only for SCE_Model_evaluation.
`Simulations`	A list containing model predictions: • For SCE: must contain 'Training', 'Validation', and 'Testing' components • For SCA: must contain 'Testing_sim' component The structure should align with the output generated by the respective model training function.
`Predictant`	A character vector specifying the name(s) of the dependent variable(s) to be evaluated (e.g., c("swvl3", "swvl4")). The specified names must exactly match those used in model training.
`digits`	An integer specifying the number of decimal places to retain when reporting evaluation metrics. Default value is 3.

Details

The functions perform comprehensive input validation:

1. Data frame structure validation
2. Presence of required components in Simulations list
3. Existence of predictants in both data and simulations
4. Matching row counts between data and simulations
5. Proper handling of NaN values and zero/negative values

The evaluation process:

1. Removes NaN values from both observed and simulated data
2. Handles zero or negative values by replacing them with 0.0001
3. Calculates all six metrics for each predictant
4. Formats the results with specified number of decimal places

Value

For SCA_Model_evaluation:

- If single predictant: Returns a data.frame with column "Testing"
- If multiple predictants: Returns a list of data.frames, one for each predictant

For SCE_Model_evaluation:

- If single predictant: Returns a data.frame with columns "Training", "Validation", and "Testing"
- If multiple predictants: Returns a list of data.frames, one for each predictant

Each data.frame contains the following metrics:

- MAE: Mean Absolute Error (mean(abs(obs - sim)))
- RMSE: Root Mean Square Error (sqrt(mean((obs - sim)^2)))
- NSE: Nash-Sutcliffe Efficiency (1 - (sum((obs - sim)^2) / sum((obs - mean(obs))^2)))
- Log.NSE: NSE calculated on log-transformed values
- R2: R-squared calculated using linear regression
- kge: Kling-Gupta Efficiency (1 - sqrt((r-1)^2 + (alpha-1)^2 + (beta-1)^2))

Author(s)

Kailong Li <lkl98509509@gmail.com>

See Also

`SCE`

`Model_simulation` *Perform Simulations using a Trained SCE Model*

Description

This function facilitates the simulation of dependent variables based on the specified independent variables utilizing a previously trained SCE model. It generates predictions for training, out-of-bag validation, and testing datasets. The function includes comprehensive input validation for data types, missing values, and predictor matching.

Usage

```
Model_simulation(Testing_data, model)
```

Arguments

`Testing_data`	A data.frame or matrix comprising the data that will be used to test the model. Must contain all the predictors used in the model. Must not contain missing values.
`model`	The trained SCE model object generated through the `SCE` function. This model contains the necessary information and parameters for conducting simulations.

Details

The simulation process involves the following steps:

1. Input validation:
   - Data type and structure checks (data.frame or matrix)
   - Missing value checks
   - Predictor matching with training data
   - Numeric data validation
2. Data preparation:
   - Conversion to matrix format
   - Initialization of prediction matrices
3. Prediction generation:
   - Training predictions using all trees
   - Out-of-bag predictions using trees not trained on each sample
   - Testing predictions using all trees

Value

A list containing three components:

- Training: A data.frame containing predictions for the training dataset
- Validation: A data.frame containing out-of-bag (OOB) predictions
- Testing: A data.frame containing predictions for the testing dataset

Author(s)

Kailong Li <lkl98509509@gmail.com>

See Also

`SCE`

`OOB_validation` *Calculate Out-of-Bag (OOB) Validation Predictions*

Description

This function calculates the out-of-bag (OOB) validation predictions for an SCE model. OOB predictions are made using only the trees that did not use a particular observation during training, providing an unbiased estimate of the model's performance.

Usage

```
OOB_validation(model)
```

Arguments

`model`	A trained SCE model object generated through the `SCE` function. The model must contain OOB information in its Tree_Info component.

Details

The OOB validation process involves:

1. Extracting OOB indices and predictions from each tree
2. Weighting the predictions based on tree weights
3. Calculating weighted means for each sample

This function is typically called internally by `Model_simulation` and is not usually called directly by users.

Value

A data.frame containing the OOB predictions for each predictant. The number of rows equals the number of training samples, and the columns correspond to the predictant variables.

Author(s)

Kailong Li <lkl98509509@gmail.com>

See Also

`SCE`

RFE_SCE *Recursive Feature Elimination for SCE Models*

Description

This function implements Recursive Feature Elimination (RFE) to identify the most important predictors for SCE models. It iteratively removes the least important predictors based on Wilks' feature importance scores and evaluates model performance. The function supports both single and multiple predictants, with comprehensive input validation and performance tracking across iterations.

Usage

```
RFE_SCE(
  Training_data,
  Testing_data,
  Predictors,
  Predictant,
  Nmin,
  Ntree,
  alpha = 0.05,
  resolution = 1000,
  step = 1,
  verbose = TRUE,
  parallel = TRUE
)
```

Arguments

`Training_data`	A data.frame containing the training data. Must include all specified predictors and predictants.
`Testing_data`	A data.frame containing the testing data. Must include all specified predictors and predictants.
`Predictors`	A character vector specifying the names of independent variables to be evaluated (e.g., c("Prcp","SRad","Tmax")). Must contain at least 2 elements.
`Predictant`	A character vector specifying the name(s) of dependent variable(s) (e.g., c("swvl3","swvl4")). Must be non-empty.
`Nmin`	Integer specifying the minimal number of samples in a leaf node for cutting.
`Ntree`	Integer specifying the number of trees in the ensemble.
`alpha`	Numeric significance level for clustering, between 0 and 1. Default value is 0.05.
`resolution`	Numeric value specifying the resolution for splitting. Default value is 1000.
`step`	Integer specifying the number of predictors to remove at each iteration. Must be between 1 and (number of predictors - number of predictants). Default value is 1.

`verbose`	A logical value indicating whether to print progress information during RFE iterations. Default value is TRUE.
`parallel`	A logical value indicating whether to use parallel processing for SCE model construction. When TRUE, uses multiple CPU cores for faster computation. When FALSE, processes trees sequentially. Default value is TRUE.

Details

The RFE process involves the following steps:

1. Input validation:
    - Data frame structure validation
    - Predictor and predictant validation
    - Step size validation
2. Initialization:
    - Set up history tracking structures
    - Initialize current predictor set
3. Main RFE loop (continues while predictors > predictants + 2):
    - Train SCE model with current predictors
    - Generate predictions using Model_simulation
    - Evaluate model using SCE_Model_evaluation
    - Store performance metrics and importance scores
    - Remove least important predictors based on Wilks' scores

The function handles:

- Single and multiple predictants
- Performance tracking across iterations
- Importance score calculation
- Step-wise predictor removal

Value

A list containing:

- summary: Data.frame with columns:
    - n_predictors: Number of predictors at each iteration
    - predictors: Comma-separated list of predictors used
- performances: List of performance evaluations for each iteration
    - For single predictant: Direct performance data.frame
    - For multiple predictants: Named list of performance data.frames
- importance_scores: List of Wilks' importance scores for each iteration

Author(s)

Kailong Li <lkl98509509@gmail.com>

Examples

```
#   # This example is computationally intensive and may take a long time to run.
#   # It is recommended to run this example on a machine with a high-performance CPU.
#
#   ## Load SCE package and the supporting packages
#   library(SCE)
#   library(parallel)
#
#   data(Streamflow_training_22var)
#   data(Streamflow_testing_22var)
#
#   # Define predictors and predictants
#   Predictors <- c(
#     "Precipitation", "Radiation", "Tmax", "Tmin", "VP",
#     "Precipitation_2Mon", "Radiation_2Mon", "Tmax_2Mon", "Tmin_2Mon", "VP_2Mon",
#     "PNA", "Nino3.4", "IPO", "PDO",
#     "PNA_lag1", "Nino3.4_lag1", "IPO_lag1", "PDO_lag1",
#     "PNA_lag2", "Nino3.4_lag2", "IPO_lag2", "PDO_lag2"
#   )
#   Predictants <- c("Flow")
#
#   # Perform RFE
#   set.seed(123)
#   result <- RFE_SCE(
#     Training_data = Streamflow_training_22var,
#     Testing_data = Streamflow_testing_22var,
#     Predictors = Predictors,
#     Predictant = Predictants,
#     Nmin = 5,
#     Ntree = 48,
#     alpha = 0.05,
#     resolution = 1000,
#     step = 3,  # Number of predictors to remove at each iteration
#     verbose = TRUE,
#     parallel = TRUE
#   )
#
#   ## Access results
#   summary <- RFE_results$summary
#   performances <- RFE_results$performances
#   importance_scores <- RFE_results$importance_scores
#
#
```

SCA *Stepwise Cluster Analysis (SCA) Model*

Description

This function implements a Stepwise Cluster Analysis (SCA) model for multivariate data analysis. The SCA model recursively partitions the data space based on Wilks' Lambda statistic, creating a tree structure that can be used for prediction. The function includes comprehensive input validation for data types, missing values, and sample size requirements, and supports both single and multiple predictants.

Usage

```
SCA(Training_data, X, Y, Nmin, alpha = 0.05, resolution = 1000, verbose = FALSE)
```

Arguments

`Training_data`	A data.frame or matrix containing the training data. Must include all specified predictors and predictants. Must not contain missing values.
`X`	A character vector specifying the names of independent variables (e.g., c("Prcp","SRad","Tmax")). Must be present in Training_data. All variables must be numeric.
`Y`	A character vector specifying the name(s) of dependent variable(s) (e.g., c("swvl3","swvl4")). Must be present in Training_data. All variables must be numeric.
`Nmin`	Integer specifying the minimal number of samples in a leaf node for cutting. Must be a positive number and less than the sample size.
`alpha`	Numeric significance level for clustering, between 0 and 1. Default value is 0.05.
`resolution`	Numeric value specifying the resolution for splitting. Controls the granularity of the search for optimal split points. Default value is 1000.
`verbose`	A logical value indicating whether to print progress information during model building. Default value is FALSE.

Details

The SCA model building process involves:

1. Input validation:
   - Data type and structure checks (data.frame or matrix)
   - Missing value checks in both predictors and predictants
   - Numeric data validation for all variables
   - Sample size requirements (must be greater than Nmin)
   - Parameter validation (alpha, Nmin, resolution)
2. Data preparation:
   - Conversion of input data to matrix format
   - Dimension checks and storage
   - Parameter initialization
3. Tree construction:
   - Recursive partitioning based on Wilks' Lambda
   - Node splitting and merging
   - Leaf node creation

Value

A list containing:

- Tree: The SCA tree structure
- Map: Mapping information for predictions
- XName: Names of predictors used
- YName: Names of predictants
- type: Mapping type (currently "mean")
- totalNodes: Total number of nodes in the tree
- leafNodes: Number of leaf nodes
- cuttingActions: Number of cutting actions performed
- mergingActions: Number of merging actions performed

Author(s)

Xiuquan Wang <xxwang@upei.ca> (original SCA) Kailong Li <lkl98509509@gmail.com> (Resolution-search-based SCA)

Examples

```
## Load SCE package
library(SCE)

## Load training and testing data
data("Streamflow_training_10var")
data("Streamflow_testing_10var")

## Define independent (x) and dependent (y) variables
Predictors <- c("Prcp","SRad","Tmax","Tmin","VP","smlt","swvl1","swvl2","swvl3","swvl4")
Predictants <- c("Flow")

## Build the SCA model
Model <- SCA(
  Training_data = Streamflow_training_10var,
  X = Predictors,
  Y = Predictants,
  Nmin = 5,
  alpha = 0.05,
  resolution = 1000
)
```

`SCA_importance` *Calculate Variable Importance for a Single SCA Tree*

Description

This function calculates the importance of independent variables in explaining the variability of dependent variables for a single Stepwise Cluster Analysis (SCA) tree using the Wilks' Lambda statistic. The importance is calculated based on the contribution of each variable to the reduction in Wilks' Lambda at each split in the tree.

For calculating importance scores across all trees in an SCE ensemble, use `Wilks_importance` instead.

Usage

```
SCA_importance(model)
```

Arguments

`model`	A single SCA tree object containing: • Tree: Tree structure with Wilks' Lambda values and split information • XName: Names of predictors used

Details

The importance calculation process involves the following steps:

1. Extract Wilks' Lambda values and split information from the tree
2. Replace negative Wilks' Lambda values with zero
3. Calculate raw importance for each split:
    - Importance = (left_samples + right_samples) / total_samples * (1 - Wilks' Lambda)
4. Aggregate importance scores by predictor
5. Normalize importance scores to sum to 1

The function handles:

- Different sets of predictors in the tree
- Missing or invalid splits
- Both single and multiple predictants
- Trees with no splits (returns NULL)

Relationship with Wilks_importance:

- `SCA_importance` calculates importance scores for a single SCA tree
- `Wilks_importance` calculates importance scores across all trees in an SCE ensemble
- Both functions use the same underlying importance calculation method
- `Wilks_importance` with OOB_weight=FALSE is equivalent to taking the median of `SCA_importance` scores across all trees

Value

A data.frame containing:

- Predictor: Names of the predictors
- Relative_Importance: Normalized importance scores (sum to 1)

Author(s)

Kailong Li <lkl98509509@gmail.com>

References

Li, Kailong, Guohe Huang, and Brian Baetz. "Development of a Wilks feature importance method with improved variable rankings for supporting hydrological inference and modelling." Hydrology and Earth System Sciences 25.9 (2021): 4947-4966.

Examples

```
## Load SCE package and the supporting packages
library(SCE)

## Load the training and testing data files
data("Streamflow_training_10var")
data("Streamflow_testing_10var")

## Define the independent (x) and dependent (y) variables
Predictors <- c("Prcp", "SRad", "Tmax", "Tmin", "VP", "smlt", "swvl1", "swvl2", "swvl3", "swvl4")
Predictants <- c("Flow")

## Build a single SCA tree
SCA_tree <- SCA(
Training_data = Streamflow_training_10var,
X = Predictors,
Y = Predictants,
Nmin = 5,
alpha = 0.05,
resolution = 1000
)

## Calculate variable importance for the single tree
Tree_importance <- SCA_importance(SCA_tree)

## Print the results
print("Single tree importance scores:")
print(Tree_importance)

## Visualize the importance scores
Importance_ranking_sorted <- Tree_importance[order(-Tree_importance$Relative_Importance), ]
barplot(
  Importance_ranking_sorted$Relative_Importance,
  names.arg = Importance_ranking_sorted$Predictor,
  las = 2, # vertical labels
  col = "skyblue",
  main = "Variable Importance (SCE)",
  ylab = "Importance",
  xlab = "Predictor"
)
```

`SCA_tree_predict` *Make Predictions Using a Single SCA Tree*

Description

This function makes predictions using a single Stepwise Cluster Analysis (SCA) tree. It traverses the tree structure based on the predictor values in the test data and returns the predicted values for the predictants. The function includes comprehensive input validation for data types, missing values, and predictor matching.

For making predictions using an entire SCE ensemble, use `Model_simulation` instead.

Usage

```
SCA_tree_predict(Testing_data, model)
```

Arguments

`Testing_data`	A data.frame or matrix containing the test data. Must include all predictors used in model training. Must not contain missing values.
`model`	A trained SCA model object returned by the SCA function.

Details

The prediction process involves the following steps:

1. Input validation:
    - Data type and structure checks (data.frame or matrix)
    - Missing value checks
    - Predictor matching with training data
    - Numeric data validation
2. Data preparation:
    - Conversion to matrix format
    - Initialization of prediction matrix
3. Tree traversal and prediction:
    - Processing each test sample through the tree
    - Generating predictions using leaf node mappings

Value

A list containing:

- A data.frame containing predictions for the test data

Author(s)

Kailong Li <lkl98509509@gmail.com>

References

Li, Kailong, Guohe Huang, and Brian Baetz. "Development of a Wilks feature importance method with improved variable rankings for supporting hydrological inference and modelling." Hydrology and Earth System Sciences 25.9 (2021): 4947-4966.

Examples

```
## Load SCE package
library(SCE)

## Load training and testing data
data("Streamflow_training_10var")
data("Streamflow_testing_10var")

## Define independent (x) and dependent (y) variables
Predictors <- c("Prcp","SRad","Tmax","Tmin","VP","smlt","swvl1","swvl2","swvl3","swvl4")
Predictants <- c("Flow")

## Build the SCA model
Model <- SCA(
Training_data = Streamflow_training_10var,
X = Predictors,
Y = Predictants,
Nmin = 5,
alpha = 0.05,
resolution = 1000
)

## Make predictions
Predictions <- SCA_tree_predict(
Testing_data = Streamflow_testing_10var,
model = Model
)
```

SCE *Build a Stepwise Clustered Ensemble (SCE) Model*

Description

This function builds a Stepwise Clustered Ensemble (SCE) model for multivariate data analysis. The SCE model is an ensemble of Stepwise Cluster Analysis (SCA) trees, where each tree is built using bootstrap samples and random feature selection. The function includes comprehensive input validation for data types, missing values, and sample size requirements.

Usage

```
SCE(Training_data, X, Y, mfeature, Nmin, Ntree,
    alpha = 0.05, resolution = 1000, verbose = FALSE, parallel = TRUE)
```

Arguments

`Training_data`	A data.frame or matrix containing the training data. Must contain all specified predictors and predictants. Must not contain missing values.
`X`	A character vector specifying the names of independent (predictor) variables (e.g., c("Prcp","SRad","Tmax")). Must be present in Training_data. All variables must be numeric.
`Y`	A character vector specifying the name(s) of dependent (predictant) variable(s) (e.g., c("Flow") or c("swvl3","swvl4")). Must be present in Training_data. All variables must be numeric.
`mfeature`	An integer specifying how many features will be randomly selected for each tree. Recommended value is round(0.5 * length(X)).
`Nmin`	An integer specifying the minimal number of samples in a leaf node for cutting. Must be greater than the number of predictants.
`Ntree`	An integer specifying how many trees (ensemble members) will be built. Recommended values range from 50 to 500 depending on data complexity.
`alpha`	Numeric significance level for clustering, between 0 and 1. Default value is 0.05.
`resolution`	Numeric value specifying the resolution for splitting. Controls the granularity of the search for optimal split points. Default value is 1000.
`verbose`	A logical value indicating whether to print progress information during model building. Default value is FALSE.
`parallel`	A logical value indicating whether to use parallel processing for tree construction. When TRUE, uses multiple CPU cores for faster computation. When FALSE, processes trees sequentially. Default value is TRUE.

Details

The SCE model is built using the following steps:

1. **Input Validation**:
   - Data type and structure checks
   - Missing value detection
   - Numeric data validation
   - Sample size requirements verification
2. **Data Preparation**:
   - Conversion to appropriate format
   - Dimension checks
   - Parameter initialization
3. **Tree Construction**:
   - Generation of bootstrap samples
   - Random feature selection for each tree
   - Parallel construction of SCA trees
4. **Model Evaluation**:
   - Calculation of out-of-bag (OOB) errors
   - Weighting of trees based on OOB performance

The ensemble approach provides improved prediction accuracy and robustness compared to single SCA trees, while the OOB validation provides unbiased performance estimates.

Value

A list containing the ensemble model with the following components:

- `Trees`: A list of SCA tree models, each containing:
  - `Tree`: The SCA tree structure
  - `Map`: Mapping information
  - `XName`: Names of predictors used
  - `YName`: Names of predictants
  - `type`: Mapping type
  - `totalNodes`: Total number of nodes
  - `leafNodes`: Number of leaf nodes
  - `cuttingActions`: Number of cutting actions
  - `mergingActions`: Number of merging actions
  - `OOB_error`: Out-of-bag R-squared error
  - `OOB_sim`: Out-of-bag predictions
  - `Sample`: Bootstrap sample indices
  - `Tree_Info`: Tree-specific information
  - `Training_data`: Training data used for the tree
  - `weight`: Tree weight based on OOB performance

Author(s)

Kailong Li <lkl98509509@gmail.com>

References

Li, Kailong, Guohe Huang, and Brian Baetz. Development of a Wilks feature importance method with improved variable rankings for supporting hydrological inference and modelling. Hydrology and Earth System Sciences 25.9 (2021): 4947-4966.

Wang, X., G. Huang, Q. Lin, X. Nie, G. Cheng, Y. Fan, Z. Li, Y. Yao, and M. Suo (2013), A stepwise cluster analysis approach for downscaled climate projection - A Canadian case study. Environmental Modelling & Software, 49, 141-151.

Huang, G. (1992). A stepwise cluster analysis method for predicting air quality in an urban environment. Atmospheric Environment (Part B. Urban Atmosphere), 26(3): 349-357.

Liu, Y. Y. and Y. L. Wang (1979). Application of stepwise cluster analysis in medical research. Scientia Sinica, 22(9): 1082-1094.

See Also

`SCA` for single tree construction, `SCE_Prediction` for making predictions, `Model_simulation` for comprehensive model evaluation, `Wilks_importance` for variable importance analysis, `RFE_SCE` for recursive feature elimination

Examples

```
## Load required packages
library(SCE)
library(parallel)

## Load example datasets
data("Streamflow_training_10var")
data("Streamflow_testing_10var")

## Define predictors and predictants
Predictors <- c("Prcp","SRad","Tmax","Tmin","VP","smlt","swvl1","swvl2","swvl3","swvl4")
Predictants <- c("Flow")

## Build the SCE model
Model <- SCE(
Training_data = Streamflow_training_10var,
X = Predictors,
Y = Predictants,
mfeature = round(0.5 * length(Predictors)),
Nmin = 5,
Ntree = 48,
alpha = 0.05,
resolution = 1000,
parallel = FALSE
)

## Generate predictions for test data
```

```
predictions <- SCE_Prediction(
X_sample = Streamflow_testing_10var,
model = Model
)

## Conduct comprehensive model evaluation
Results <- Model_simulation(
Testing_data = Streamflow_testing_10var,
model = Model
)

## Access different prediction components
training_predictions <- Results$Training
validation_predictions <- Results$Validation
testing_predictions <- Results$Testing

## Calculate variable importance with OOB weighting (default)
Importance_weighted <- Wilks_importance(Model)

## Calculate variable importance without OOB weighting
Importance_unweighted <- Wilks_importance(Model, OOB_weight = FALSE)

## Visualize the importance scores
Importance_ranking_sorted <- Importance_weighted[
order(-Importance_weighted$Relative_Importance),
]
barplot(
Importance_ranking_sorted$Relative_Importance,
names.arg = Importance_ranking_sorted$Predictor,
las = 2,
col = "skyblue",
main = "Variable Importance (SCE)",
ylab = "Importance",
xlab = "Predictor"
)
```

SCE_Prediction *Generate Predictions using an SCE Model*

Description

This function generates predictions for new data using a trained SCE model. It combines predictions from individual trees in the ensemble, weighted by their respective importance weights.

Usage

```
SCE_Prediction(X_sample, model)
```

Arguments

`X_sample`	A data.frame or matrix containing the predictor variables for which predictions are to be made. Must contain all predictors used in model training.
`model`	A trained SCE model object generated through the `SCE` function. The model must contain the trained trees and their weights.

Details

The prediction process involves:

1. Generating predictions from each tree in the ensemble
2. Weighting the predictions based on tree weights
3. Combining the weighted predictions to form the ensemble prediction

This function is typically called internally by `Model_simulation` and is not usually called directly by users.

Value

A matrix containing the ensemble predictions for each predictant. The number of rows equals the number of samples in X_sample, and the columns correspond to the predictant variables.

Author(s)

Kailong Li <lkl98509509@gmail.com>

See Also

`SCE`

`Streamflow_testing_10var` *Streamflow Testing Dataset with 10 Variables*

Description

A dataset containing streamflow and related environmental variables for testing purposes. This dataset is used in the examples to demonstrate the SCE package functionality.

Usage

```
data("Streamflow_testing_10var")
```

Format

A data frame with the following variables:

Flow	Streamflow measurements
Prcp	Precipitation
SRad	Solar radiation
Tmax	Maximum temperature
Tmin	Minimum temperature
VP	Vapor pressure
X	Index variable
smlt	Snow melt
swvl1	Soil water volume layer 1
swvl2	Soil water volume layer 2
swvl3	Soil water volume layer 3
swvl4	Soil water volume layer 4

Source

The data was collected from environmental monitoring stations.

Examples

```
data("Streamflow_testing_10var")
str(Streamflow_testing_10var)
```

`Streamflow_testing_22var`
Streamflow Testing Dataset with 22 Variables

Description

A dataset containing streamflow and related environmental variables for testing purposes. This dataset includes additional climate indices and their lagged values.

Usage

```
data("Streamflow_testing_22var")
```

Format

A data frame with the following variables:

Flow	Streamflow measurements
IPO	Interdecadal Pacific Oscillation
IPO_lag1	IPO with 1-month lag
IPO_lag2	IPO with 2-month lag
Nino3.4	Nino 3.4 index
Nino3.4_lag1	Nino 3.4 with 1-month lag
Nino3.4_lag2	Nino 3.4 with 2-month lag
PDO	Pacific Decadal Oscillation
PDO_lag1	PDO with 1-month lag
PDO_lag2	PDO with 2-month lag
PNA	Pacific North American pattern
PNA_lag1	PNA with 1-month lag
PNA_lag2	PNA with 2-month lag
Precipitation	Monthly precipitation
Precipitation_2Mon	2-month precipitation
Radiation	Solar radiation
Radiation_2Mon	2-month solar radiation
Tmax	Maximum temperature
Tmax_2Mon	2-month maximum temperature
Tmin	Minimum temperature
Tmin_2Mon	2-month minimum temperature
VP	Vapor pressure
VP_2Mon	2-month vapor pressure

Source

The data was collected from environmental monitoring stations and climate indices databases.

Examples

```
data("Streamflow_testing_22var")
str(Streamflow_testing_22var)
```

Streamflow_training_10var *Example Streamflow Training Dataset for the SCE Model (10 variables)*

Description

The "Streamflow_training_10var" dataset serves as an example dataset to illustrate the functioning and application of the SCE model within the SCE package. It contains various environmental variables measured on a monthly scale, which are used as inputs in the SCE model for model training.

Usage

```
data("Streamflow_training_10var")
```

Format

A data frame with a number of rows (X) and columns (Y) containing the following variables:

Date The date and time of the data point.

Prcp The monthly mean daily precipitation measured in millimeters (mm). This data is derived from the Daymet dataset.

SRad The monthly mean daily short-wave solar radiation measured in Watts per square meter (W/m^2), sourced from the Daymet dataset.

Tmax The monthly mean daily maximal temperature recorded in degrees Celsius, taken from the Daymet dataset.

Tmin The monthly mean daily minimal temperature recorded in degrees Celsius, also derived from the Daymet dataset.

VP The monthly mean daily vapor pressure measured in Pascals (Pa), obtained from the Daymet dataset.

smlt The sum of monthly snowmelt measurements in meters (m), taken from the ERA5 land dataset.

swvl1 The volumetric soil water content in layer 1 measured in cubic meters per cubic meter (m^3/m^3), sourced from the ERA5 land dataset.

swvl2 The volumetric soil water content in layer 2, measured similarly to swvl1, sourced from the ERA5 land dataset.

swvl3 The volumetric soil water content in layer 3, measured similarly to swvl1, sourced from the ERA5 land dataset.

swvl4 The volumetric soil water content in layer 4, measured similarly to swvl1, sourced from the ERA5 land dataset.

Flow The monthly mean daily streamflow rate measured in cubic feet per second (cfs), provided by the United States Geological Survey (USGS).

Source

The data is compiled from various recognized sources including:

- ERA5 Land: A global land-surface dataset at 9km resolution, available from the Copernicus Climate Change Service.
- Daymet Version 4: Daily Surface Weather and Climatological Summaries.
- United States Geological Survey (USGS): A scientific agency of the United States government that studies natural resources, natural hazards, and the landscape of the United States.

Examples

```
# Load the "Streamflow_training_10var" dataset and display the first few rows
data(Streamflow_training_10var)
head(Streamflow_training_10var)
```

`Streamflow_training_22var` *Streamflow Training Dataset with 22 Variables*

Description

A dataset containing streamflow and related environmental variables for training purposes. This dataset includes additional climate indices and their lagged values.

Usage

```
data("Streamflow_training_22var")
```

Format

A data frame with the following variables:

Flow Streamflow measurements

IPO Interdecadal Pacific Oscillation

IPO_lag1 IPO with 1-month lag

IPO_lag2 IPO with 2-month lag

Nino3.4 Nino 3.4 index

Nino3.4_lag1 Nino 3.4 with 1-month lag

Nino3.4_lag2 Nino 3.4 with 2-month lag

PDO Pacific Decadal Oscillation

PDO_lag1 PDO with 1-month lag

PDO_lag2 PDO with 2-month lag

PNA Pacific North American pattern

PNA_lag1 PNA with 1-month lag

PNA_lag2	PNA with 2-month lag
Precipitation	Monthly precipitation
Precipitation_2Mon	2-month precipitation
Radiation	Solar radiation
Radiation_2Mon	2-month solar radiation
Tmax	Maximum temperature
Tmax_2Mon	2-month maximum temperature
Tmin	Minimum temperature
Tmin_2Mon	2-month minimum temperature
VP	Vapor pressure
VP_2Mon	2-month vapor pressure

Source

The data was collected from environmental monitoring stations and climate indices databases.

Examples

```
data("Streamflow_training_22var")
str(Streamflow_training_22var)
```

`Wilks_importance` *Calculate Variable Importance using Wilks' Lambda*

Description

This function calculates the importance of independent variables in explaining the variability of dependent variables using the Wilks' Lambda statistic. The importance is calculated based on the contribution of each variable to the reduction in Wilks' Lambda at each split in the SCA trees. The function supports both unweighted and OOB-weighted importance calculations.

For calculating importance scores for a single SCA tree, use `SCA_importance` instead.

Usage

```
Wilks_importance(model, OOB_weight = TRUE)
```

Arguments

`model` A trained SCE model object containing a list of SCA trees. Each tree should contain:

- Tree: Tree structure with Wilks' Lambda values and split information
- XName: Names of predictors used
- weight: Tree weight (if OOB_weight = TRUE)

`OOB_weight`	A logical value indicating whether to weight the importance scores by the tree's OOB performance. • If TRUE (default): Importance scores are weighted by each tree's OOB performance • If FALSE: Importance scores are calculated using the median across trees

Details

The importance calculation process involves the following steps:

1. Extract Wilks' Lambda values and split information from each tree
2. Replace negative Wilks' Lambda values with zero
3. Calculate raw importance for each split:
   - Importance = (left_samples + right_samples) / total_samples * (1 - Wilks' Lambda)
4. Aggregate importance scores by predictor:
   - If OOB_weight = TRUE: Weight by tree's OOB performance and sum
   - If OOB_weight = FALSE: Take median across trees
5. Normalize importance scores to sum to 1

The function handles:

- Multiple trees in the ensemble
- Different sets of predictors in each tree
- Missing or invalid splits
- Both single and multiple predictants
- Trees with no splits (returns NULL for those trees)

Relationship with SCA_importance:

- `Wilks_importance` calculates importance scores across all trees in an SCE ensemble
- `SCA_importance` calculates importance scores for a single SCA tree
- Both functions use the same underlying importance calculation method
- `Wilks_importance` with OOB_weight=FALSE is equivalent to taking the median of `SCA_importance` scores across all trees

Value

A data.frame containing:

- Predictor: Names of the predictors
- Relative_Importance: Normalized importance scores (sum to 1)

Author(s)

Kailong Li <lkl98509509@gmail.com>

References

Li, Kailong, Guohe Huang, and Brian Baetz. "Development of a Wilks feature importance method with improved variable rankings for supporting hydrological inference and modelling." Hydrology and Earth System Sciences 25.9 (2021): 4947-4966.

See Also

`SCE`

Index

Air_quality_testing, 2
Air_quality_training, 3

Model_evaluation, 4
Model_simulation, 5, *15*, *19*

OOB_validation, 7

RFE_SCE, 8, *19*

SCA, 10, *19*
SCA_importance, 13, *26, 27*
SCA_Model_evaluation
        *(*Model_evaluation*)*, 4
SCA_tree_predict, 15
SCE, *5–7*, 17, *21*, *28*
SCE_Model_evaluation
        *(*Model_evaluation*)*, 4
SCE_Prediction, *19*, 20
Streamflow_testing_10var, 21
Streamflow_testing_22var, 22
Streamflow_training_10var, 24
Streamflow_training_22var, 25

Wilks_importance, *13*, *19*, 26