Package 'SCE'

July 2, 2025

Title Stepwise Clustered Ensemble

Version 1.1.0

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 2025-07-02 07:00:02 UTC

Contents

`Air_quality_training` *Air Quality Datasets*

Description

These datasets contain air quality measurements for training and testing purposes. They include various air pollutant concentrations and meteorological variables measured at different locations and times.

Usage

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

Format

Both datasets are data frames with 8760 rows and 12 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)

Details

Dataset Differences:

- `Air_quality_training`: Used for training SCA and SCE models
- `Air_quality_testing`: Used for testing trained models

Variable Descriptions:

- **PM2.5, PM10**: Particulate matter concentrations, important indicators of air quality
- **SO2, NO2, CO, O3**: Major air pollutants regulated by environmental agencies
- **TEMP, PRES, DEWP**: Meteorological variables affecting air quality
- **RAIN, WSPM**: Weather conditions that influence pollutant dispersion

Source

Air quality monitoring stations

`evaluate` *Model Evaluation*

Description

Functions for evaluating model performance using comprehensive metrics. The package provides both generic S3 methods and direct function calls for model evaluation.

Usage

```
evaluate(object, ...)

## S3 method for class 'SCA'
evaluate(object, Testing_data, Predictant, digits = 3, ...)

## S3 method for class 'SCE'
evaluate(object, Testing_data, Training_data, Predictant, digits = 3, ...)

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

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

Arguments

`object` An object for which performance should be evaluated.

`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 objects and `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.

`...` Additional arguments passed to methods.

Details

Evaluation Metrics:

The functions evaluate model performance using six distinct metrics:

1. **MAE (Mean Absolute Error)**: Average absolute difference between observed and predicted values
2. **RMSE (Root Mean Square Error)**: Square root of the average squared differences
3. **NSE (Nash-Sutcliffe Efficiency)**: Measures the relative magnitude of residual variance compared to observed variance
4. **Log.NSE**: NSE calculated on log-transformed values for better handling of skewed distributions
5. **R2 (R-squared)**: Coefficient of determination from linear regression
6. **KGE (Kling-Gupta Efficiency)**: Combines correlation, bias, and variability ratio

Function Differences:

- `evaluate.SCA()`: S3 method for single SCA trees (calls `SCA_Model_evaluation`)
- `evaluate.SCE()`: S3 method for SCE ensembles (calls `SCE_Model_evaluation`)
- `SCA_Model_evaluation()`: Direct function for SCA model evaluation
- `SCE_Model_evaluation()`: Direct function for SCE model evaluation

Input Validation: 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

Data Processing:

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 models and `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 models and `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

SCA, SCE

---

importance *Variable Importance Analysis*

Description

Functions for calculating variable importance scores using Wilks' Lambda method. The package provides both generic S3 methods and direct function calls for importance analysis.

Usage

```
importance(object, ...)

## S3 method for class 'SCA'
importance(object, ...)

## S3 method for class 'SCE'
importance(object, OOB_weight = TRUE, ...)

Wilks_importance(model, OOB_weight = TRUE)

SCA_importance(model)
```

Arguments

`object` An object for which importance scores should be calculated.

`model` A trained model object:

- For `Wilks_importance`: SCE model object (S3 class "SCE") containing a list of SCA trees
- For `SCA_importance`: Single SCA tree object (S3 class "SCA")

`OOB_weight` Logical indicating whether to use out-of-bag weighting for importance calculation. Default is TRUE. Only used for SCE objects and `Wilks_importance`.

`...` Additional arguments passed to methods.

Details

Importance Calculation Method:

All functions use the Wilks' Lambda statistic to calculate variable importance:

1. Extract Wilks' Lambda values and split information from tree(s)
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

Function Differences:

- `importance.SCA()`: S3 method for single SCA trees
- `importance.SCE()`: S3 method for SCE ensembles (calls `Wilks_importance`)
- `Wilks_importance()`: Direct function for SCE ensembles with OOB weighting options
- `SCA_importance()`: Direct function for single SCA trees

OOB Weighting:

- If `OOB_weight = TRUE`: Importance scores are weighted by each tree's OOB performance
- If `OOB_weight = FALSE`: Importance scores are calculated using the median across 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

SCA, SCE

---

predict.SCA *Model Prediction and Simulation*

Description

Functions for making predictions and performing simulations using trained SCA and SCE models. The package provides both S3 methods and direct function calls for various prediction scenarios.

Usage

```
## S3 method for class 'SCA'
predict(object, newdata, ...)

## S3 method for class 'SCE'
predict(object, newdata, ...)

Model_simulation(model, Testing_data)

SCA_tree_predict(model, Testing_data)

SCE_Prediction(X_sample, model)

OOB_validation(model)
```

Arguments

`object` An object for which predictions should be made.

`newdata` A data.frame or matrix containing new data for prediction. Must contain the same predictor variables as used in training.

`model` A trained model object:

- For `Model_simulation`: SCE model object (S3 class "SCE")
- For `SCA_tree_predict`: SCA model object (S3 class "SCA")
- For `SCE_Prediction`: SCE model object (S3 class "SCE")
- For `OOB_validation`: SCE model object (S3 class "SCE")

`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.

`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.

`...` Additional arguments passed to methods.

Details

Prediction Methods:

- `predict.SCA()`: S3 method for single SCA trees (calls `SCA_tree_predict`)
- `predict.SCE()`: S3 method for SCE ensembles (calls `Model_simulation`)
- `Model_simulation()`: Comprehensive simulation for SCE models with training, validation, and testing predictions
- `SCA_tree_predict()`: Direct function for single SCA tree predictions
- `SCE_Prediction()`: Direct function for SCE ensemble predictions
- `OOB_validation()`: Internal function for calculating out-of-bag predictions

Prediction Process:

For SCA models:

1. Input validation (data types, missing values, predictor matching)
2. Data preparation (conversion to matrix format)
3. Tree traversal and prediction using leaf node mappings

For SCE models:

1. Input validation (data types, missing values, predictor matching)
2. Data preparation (conversion to matrix format)
3. Training predictions using all trees
4. Out-of-bag predictions using trees not trained on each sample
5. Testing predictions using all trees
6. Weighting predictions based on tree weights

Out-of-Bag (OOB) Validation:

- OOB predictions are made using only trees that did not use a particular observation during training
- Provides unbiased estimate of model performance
- Used internally by `Model_simulation` for validation predictions

Input Validation: All functions perform comprehensive validation:

1. Data type and structure checks (data.frame or matrix)
2. Missing value checks
3. Predictor matching with training data
4. Numeric data validation

Value

For S3 methods:

- `predict.SCA()`: A matrix of predicted values for the predictant variables
- `predict.SCE()`: A list containing Training, Validation, and Testing predictions

For direct functions:

- `Model_simulation()`: A list containing three components:
  - Training: Predictions for the training dataset
  - Validation: Out-of-bag (OOB) predictions
  - Testing: Predictions for the testing dataset
- `SCA_tree_predict()`: A list containing predictions for the test data
- `SCE_Prediction()`: A matrix containing ensemble predictions for each predictant
- `OOB_validation()`: A data.frame containing OOB predictions for each predictant

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

`SCA`, `SCE`

`print.SCA` *Print and Summary Methods for SCA and SCE Objects*

Description

Methods for printing and summarizing SCA (Stepwise Cluster Analysis) and SCE (Stepwise Clustered Ensemble) objects.

Usage

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

## S3 method for class 'SCA'
summary(object, ...)

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

## S3 method for class 'SCE'
summary(object, ...)
```

Arguments

`x, object` An SCA or SCE object returned by the `SCA()` or `SCE()` function.

`...` Additional arguments passed to methods.

Value

`print.SCA` and `print.SCE` return the object invisibly. `summary.SCA` returns a summary of the SCA model including tree statistics. `summary.SCE` returns a summary of the SCE model including ensemble statistics.

See Also

`SCA`, `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.

The package also provides a `Plot_RFE` function for visualizing RFE results, showing validation and testing R2 values as a function of the number of predictors.

Usage

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

Plot_RFE(
  rfe_result,
  main = "Validation and Testing R2 vs Number of Predictors",
  col_validation = "blue",
  col_testing = "red",
  pch = 16,
  lwd = 2,
  cex = 1.2,
  legend_pos = "bottomleft",
  ...
)
```

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.

Plot_RFE Arguments:

`rfe_result` The result object from RFE_SCE function containing summary and performances components.

`main` Title for the plot. Default is "Validation and Testing R2 vs Number of Predictors".

`col_validation` Color for validation line. Default is "blue".

`col_testing` Color for testing line. Default is "red".

`pch` Point character for markers. Default is 16 (filled circle).

`lwd` Line width. Default is 2.

`cex` Point size. Default is 1.2.

`legend_pos` Position of legend. Default is "bottomleft".

`...` Additional arguments passed to plot function.

Details

RFE_SCE Process: 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

Plot_RFE Function: Creates a base R plot showing validation and testing R2 values as a function of the number of predictors during the RFE process. The function:

- Extracts R2 values from RFE results
- Converts formatted strings to numeric values
- Creates a line plot with points and lines
- Includes a legend distinguishing validation and testing performance
- Supports customization of colors, line styles, and plot appearance
- Uses only base R graphics (no external dependencies)

Value

RFE_SCE: 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

Plot_RFE: Invisibly returns a list containing:

- n_predictors: Vector of predictor counts
- validation_r2: Vector of validation R2 values
- testing_r2: Vector of testing R2 values

Author(s)

Kailong Li <lkl98509509@gmail.com>

See Also

See the generic functions `importance` and `evaluate` for SCE objects. For visualization of RFE results, see `Plot_RFE`.

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 <- result$summary
#   performances <- result$performances
#   importance_scores <- result$importance_scores
#
#   ## Plot RFE results
#   Plot_RFE(result)
#
#   ## Customized plot
#   Plot_RFE(result,
#           main = "My RFE Results",
#           col_validation = "darkblue",
#           col_testing = "darkred",
#           lwd = 3,
#           cex = 1.5)
#
#   ## Note: The RFE_SCE function internally uses S3 methods for SCE models
#   ## including importance() and evaluate() for model analysis
#
#
```

SCE *Stepwise Clustered Ensemble (SCE) and Stepwise Cluster Analysis (SCA) Models*

Description

This package provides two main modeling approaches:

SCA (Stepwise Cluster Analysis): A single tree model that recursively partitions the data space based on Wilks' Lambda statistic, creating a tree structure for prediction.

SCE (Stepwise Clustered Ensemble): An ensemble of SCA trees built using bootstrap samples and random feature selection, providing improved prediction accuracy and robustness.

Both functions include comprehensive input validation for data types, missing values, and sample size requirements, and support both single and multiple predictants.

Usage

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

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 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.

`mfeature` An integer specifying how many features will be randomly selected for each tree. Recommended value is round(0.5 * length(X)). Only used for SCE.

`Ntree` An integer specifying how many trees (ensemble members) will be built. Recommended values range from 50 to 500 depending on data complexity. Only used for SCE.

`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. Only used for SCE.

Details

Model Building Process:

SCA (Single Tree):

1. Input validation (data types, missing values, sample size requirements)
2. Data preparation (conversion to matrix format, parameter initialization)
3. Tree construction (recursive partitioning based on Wilks' Lambda)

SCE (Ensemble):

1. Input validation (data types, missing values, sample size requirements)
2. Data preparation (conversion to matrix format, parameter initialization)

3. Tree construction (bootstrap samples, random feature selection, parallel SCA tree building)
4. Model evaluation (OOB error calculation, tree weighting)

Key Differences:

- **SCA**: Single tree, deterministic, faster training, potentially less robust
- **SCE**: Multiple trees, ensemble approach, improved accuracy, OOB validation, parallel processing

When to Use:

- **SCA**: Quick exploration, simple relationships, limited computational resources
- **SCE**: Production models, complex relationships, when accuracy is critical

Value

For SCA: An S3 object of class "SCA" 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
- call: Function call

For SCE: An S3 object of class "SCE" 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
- `predictors`: Names of predictor variables
- `predictants`: Names of predictant variables
- `parameters`: Model parameters
- `call`: Function call

Both objects support S3 methods: `print()`, `summary()`, `predict()`, `importance()`, and `evaluate()`.

Author(s)

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

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

`predict`, `importance`, `evaluate` for S3 methods, `RFE_SCE` for recursive feature elimination

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")

## Example 1: Build SCA model (single tree)
sca_model <- SCA(
Training_data = Streamflow_training_10var,
X = Predictors,
Y = Predictants,
Nmin = 5,
alpha = 0.05,
resolution = 1000
)

## Use S3 methods for SCA model inspection
print(sca_model)
summary(sca_model)

## Make predictions using S3 method
sca_predictions <- predict(sca_model, Streamflow_testing_10var)

## Calculate variable importance using S3 method
sca_importance <- importance(sca_model)

## Evaluate SCA model performance using S3 method
sca_evaluation <- evaluate(
object = sca_model,
Testing_data = Streamflow_testing_10var,
Predictant = Predictants
)

## Example 2: Build SCE model (ensemble)
sce_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
)

## Use S3 methods for SCE model inspection
print(sce_model)
summary(sce_model)

## Generate predictions using S3 method
sce_predictions <- predict(sce_model, Streamflow_testing_10var)

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

## Calculate variable importance using S3 method
sce_importance <- importance(sce_model)

## Evaluate SCE model performance using S3 method
```

```
sce_evaluation <- evaluate(
object = sce_model,
Testing_data = Streamflow_testing_10var,
Training_data = Streamflow_training_10var,
Predictant = Predictants
)
```

`Streamflow_training_10var`

Streamflow Datasets

Description

These datasets contain streamflow and related environmental variables for training and testing purposes. They are used in examples to demonstrate the SCE package functionality with different levels of complexity.

Usage

```
data("Streamflow_training_10var")
data("Streamflow_training_22var")
data("Streamflow_testing_10var")
data("Streamflow_testing_22var")
```

Format

Streamflow_training_10var: A data frame with basic environmental variables:

Date The date and time of the data point

Prcp The monthly mean daily precipitation measured in millimeters (mm), 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)

Streamflow_training_22var: A data frame with extended variables including climate indices:

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

Streamflow_testing_10var: A data frame with basic environmental variables (same structure as training):

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

Streamflow_testing_22var: A data frame with extended variables including climate indices (same structure as training):

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

Details

Dataset Categories:

- **Training Datasets**: Used for building SCA and SCE models
  - `Streamflow_training_10var`: Basic dataset with 12 variables, suitable for introductory examples
  - `Streamflow_training_22var`: Extended dataset with 24 variables, includes climate indices and lagged values
- **Testing Datasets**: Used for evaluating trained models
  - `Streamflow_testing_10var`: Basic dataset with 12 variables, matches training structure
  - `Streamflow_testing_22var`: Extended dataset with 24 variables, matches training structure

Variable Categories:

- **Hydrological**: Flow, Precipitation, Snow melt, Soil water volumes
- **Meteorological**: Temperature (max/min), Solar radiation, Vapor pressure
- **Climate Indices**: IPO, Nino3.4, PDO, PNA (with lagged versions)
- **Time Aggregations**: 2-month averages for key variables

Climate Indices:

- **IPO**: Interdecadal Pacific Oscillation - long-term climate pattern
- **Nino3.4**: El Niño-Southern Oscillation index
- **PDO**: Pacific Decadal Oscillation - long-term ocean temperature pattern
- **PNA**: Pacific North American pattern - atmospheric circulation pattern

Data Sources: 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
- Climate indices databases for the extended datasets

Source

Environmental monitoring stations, climate indices databases, ERA5 Land, Daymet, and USGS

Index