Package 'SVEMnet'

December 21, 2024

Type Package

Title Self-Validated Ensemble Models with Elastic Net Regression

Version 1.3.0

Date 2024-12-21

Maintainer Andrew T. Karl <akarl@asu.edu>

Description Implements Self-Validated Ensemble Models (SVEM, Lemkus et al. (2021) <doi:10.1016/j.chemolab.2021.104439>) using Elastic Net regression via 'glmnet' (Friedman et al. <doi:10.18637/jss.v033.i01>). SVEM averages predictions from multiple models fitted to fractionally weighted bootstraps of the data, tuned with anti-correlated validation weights. Also implements the randomized permutation whole model test for SVEM (Karl (2024) <doi:10.1016/j.chemolab.2024.105122>). \\Code for the whole model test was plementary material of Karl (2024). Development of this package was assisted by 'GPT o1-preview' for code structure and documentation.

Depends R (>= 3.5.0)

Imports glmnet, stats, gamlss, gamlss.dist, ggplot2, lhs, doParallel, parallel, foreach

VignetteBuilder knitr

Suggests knitr, rmarkdown

License GPL-2 | GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Andrew T. Karl [cre, aut] (<https://orcid.org/0000-0002-5933-8706>)

Repository CRAN

Date/Publication 2024-12-21 19:10:02 UTC

Contents

`SVEMnet-package` *SVEMnet: Self-Validated Ensemble Models with Elastic Net Regression*

Description

The `SVEMnet` package implements Self-Validated Ensemble Models (SVEM) using Elastic Net (including lasso and ridge) regression via `glmnet`. SVEM averages predictions from multiple models fitted to fractionally weighted bootstraps of the data, tuned with anti-correlated validation weights.

Functions

`SVEMnet` Fit an SVEMnet model using Elastic Net regression.

`svem_significance_test` Perform a whole-model significance test for SVEM models.

`svem_significance_test_parallel` Perform a whole-model significance test for SVEM models. Parallelized version.

`predict.svem_model` Predict method for SVEM models.

`plot.svem_model` Plot method for SVEM models.

`coef.svem_model` Plot method for SVEM models.

`glmnet_with_cv` Wrapper for cv.glmnet

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024) (it was handwritten by that author).

Author(s)

Maintainer: Andrew T. Karl <akarl@asu.edu> (ORCID)

References

Gotwalt, C., & Ramsey, P. (2018). Model Validation Strategies for Designed Experiments Using Bootstrapping Techniques With Applications to Biopharmaceuticals. *JMP Discovery Conference*. https://community.jmp.com/t5/Discovery-Summit-2018/Model-Validation-Strategies-for-Designed-Experi
ta-p/73730

Karl, A. T. (2024). A randomized permutation whole-model test heuristic for Self-Validated Ensemble Models (SVEM). *Chemometrics and Intelligent Laboratory Systems*, *249*, 105122. doi:10.1016/j.chemolab.2024.105122

Karl, A., Wisnowski, J., & Rushing, H. (2022). JMP Pro 17 Remedies for Practical Struggles with Mixture Experiments. JMP Discovery Conference. doi:10.13140/RG.2.2.34598.40003/1

Lemkus, T., Gotwalt, C., Ramsey, P., & Weese, M. L. (2021). Self-Validated Ensemble Models for Design of Experiments. *Chemometrics and Intelligent Laboratory Systems*, 219, 104439. doi:10.1016/j.chemolab.2021.104439

Xu, L., Gotwalt, C., Hong, Y., King, C. B., & Meeker, W. Q. (2020). Applications of the Fractional-Random-Weight Bootstrap. *The American Statistician*, 74(4), 345–358. doi:10.1080/00031305.2020.1731599

Ramsey, P., Gaudard, M., & Levin, W. (2021). Accelerating Innovation with Space Filling Mixture Designs, Neural Networks and SVEM. *JMP Discovery Conference*. https://community.jmp.com/t5/Abstracts/Accelerating-Innovation-with-Space-Filling-Mixture-Designs/ev-p/756841

Ramsey, P., & Gotwalt, C. (2018). Model Validation Strategies for Designed Experiments Using Bootstrapping Techniques With Applications to Biopharmaceuticals. *JMP Discovery Conference - Europe*. https://community.jmp.com/t5/Discovery-Summit-Europe-2018/Model-Validation-Strategies-for-De
ta-p/51286

Ramsey, P., Levin, W., Lemkus, T., & Gotwalt, C. (2021). SVEM: A Paradigm Shift in Design and Analysis of Experiments. *JMP Discovery Conference - Europe*. https://community.jmp.com/t5/Abstracts/SVEM-A-Paradigm-Shift-in-Design-and-Analysis-of-Experiments-2021/ev-p/756634

Ramsey, P., & McNeill, P. (2023). CMC, SVEM, Neural Networks, DOE, and Complexity: It's All About Prediction. *JMP Discovery Conference*.

`coef.svem_model` *Plot Coefficient Nonzero Percentages from a SVEMnet Model*

Description

This function calculates the percentage of bootstrap iterations in which each coefficient is nonzero.

Usage

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

Arguments

`object`	An object of class `svem_model` returned by the `SVEMnet` function.
`...`	other arguments to pass.

Value

Invisibly returns a data frame containing the percentage of bootstraps where each coefficient is nonzero.

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024) (it was handwritten by that author).

`glmnet_with_cv` *Fit a glmnet Model with Cross-Validation*

Description

A wrapper function for `cv.glmnet` that takes input arguments in a manner similar to `SVEMnet`. This function searches over multiple `alpha` values by running `cv.glmnet()` for each provided `alpha`, and then selects the combination of `alpha` and `lambda` with the best cross-validation performance.

Usage

```
glmnet_with_cv(
  formula,
  data,
  glmnet_alpha = c(0, 0.5, 1),
  standardize = TRUE,
  nfolds = 10,
  ...
)
```

Arguments

`formula`	A formula specifying the model to be fitted.
`data`	A data frame containing the variables in the model.
`glmnet_alpha`	Elastic Net mixing parameter(s) (default is `c(0, 0.5, 1)`). If multiple values are provided, `cv.glmnet` is run for each `alpha`, and the model with the lowest cross-validation error is selected.
`standardize`	Logical flag passed to `glmnet`. If `TRUE` (default), each variable is standardized before model fitting.
`nfolds`	Number of cross-validation folds (default is `10`).
`...`	Additional arguments passed to `cv.glmnet`.

Details

This function uses cv.glmnet to fit a generalized linear model with elastic net regularization, performing k-fold cross-validation to select the regularization parameter lambda. If multiple alpha values are provided, it selects the best-performing alpha-lambda pair based on the minimal cross-validation error.

After fitting, the function calculates a debiasing linear model (if possible). This is done by regressing the actual responses on the fitted values obtained from the selected model. The resulting linear model is stored in debias_fit.

Value

A list containing:

- parms: Coefficients from the selected cv.glmnet model at lambda.min.
- debias_fit: A linear model of the form y ~ y_pred used for debiasing (if applicable).
- glmnet_alpha: The vector of alpha values considered.
- best_alpha: The selected alpha value that gave the best cross-validation result.
- best_lambda: The lambda value chosen by cross-validation at the selected alpha.
- actual_y: The response vector used in the model.
- training_X: The predictor matrix used in the model.
- y_pred: The fitted values from the final model (no debiasing).
- y_pred_debiased: Debiased fitted values if debias_fit is available.
- formula: The formula used for model fitting.
- terms: The terms object extracted from the model frame.

References

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1-22. doi:10.18637/jss.v033.i01

See Also

glmnet, cv.glmnet, SVEMnet

Examples

```
set.seed(0)
n <- 50
X1 <- runif(n)
X2 <- runif(n)
y <- 1 + 2*X1 + 3*X2 + rnorm(n)
data <- data.frame(y, X1, X2)

model_cv <- glmnet_with_cv(y ~ X1 + X2, data = data, glmnet_alpha = c(0,0.5,1))
predictions <- predict_cv(model_cv, data)
```

`plot.svem_model` *Plot Method for SVEM Models*

Description

Plots actual versus predicted values for an `svem_model` using `ggplot2`.

Usage

```
## S3 method for class 'svem_model'
plot(x, plot_debiased = FALSE, ...)
```

Arguments

`x`	An object of class `svem_model`.
`plot_debiased`	Logical; if `TRUE`, includes debiased predictions if available (default is `FALSE`).
`...`	Additional arguments passed to `ggplot2` functions.

Details

This function creates an actual vs. predicted plot for the SVEM model. If `plot_debiased` is `TRUE` and debiased predictions are available, it includes them in the plot.

Plot Features:

- **Actual vs. Predicted Points:** Plots the actual response values against the predicted values from the SVEM model.
- **Debiased Predictions:** If available and `plot_debiased` is `TRUE`, debiased predictions are included.
- **Ideal Fit Line:** A dashed line representing perfect prediction (slope = 1, intercept = 0) is included for reference.

Value

A `ggplot` object showing actual versus predicted values.

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024) (it was handwritten by that author).

`plot.svem_significance_test` *Plot SVEM Significance Test Results for Multiple Responses*

Description

Plots the Mahalanobis distances for the original and permuted data from multiple SVEM significance test results.

Usage

```
## S3 method for class 'svem_significance_test'
plot(..., labels = NULL)
```

Arguments

`...`	One or more objects of class `svem_significance_test`, which are the outputs from `svem_significance_test`.
`labels`	Optional character vector of labels for the responses. If not provided, the function uses the response variable names.

Details

This function creates a combined plot of the Mahalanobis distances (`d_Y` and `d_pi_Y`) for the original and permuted data from multiple SVEM significance test results. It groups the data by response and source type, displaying original and permutation distances side by side for each response.

Usage Notes:

- Use this function to compare the significance test results across multiple responses.
- The plot shows original and permutation distances next to each other for each response.

Value

A `ggplot` object showing the distributions of Mahalanobis distances for all responses.

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024) (it was handwritten by that author).

`predict.svem_model` *Predict Method for SVEM Models*

Description

Generates predictions from a fitted `svem_model`.

Usage

```
## S3 method for class 'svem_model'
predict(object, newdata, debias = FALSE, se.fit = FALSE, ...)
```

Arguments

`object`	An object of class `svem_model`.
`newdata`	A data frame of new predictor values.
`debias`	Logical; default is `FALSE`.
`se.fit`	Logical; if `TRUE`, returns standard errors (default is `FALSE`).
`...`	Additional arguments.

Details

A debiased fit is available (along with the standard fit). This is provided to allow the user to match the output of JMP.\ https://www.jmp.com/support/help/en/18.1/?utm_source=help&utm_medium=redirect#page/jmp/overvie of-selfvalidated-ensemble-models.shtml. The debiasing coefficients are always calculated by SVEM-net(), and the predict() function determines whether the raw or debiased predictions are returned via the `debias` argument. Default is `FALSE` based on performance on unpublished simulation studies.

Value

Predictions or a list containing predictions and standard errors.

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024) (it was handwritten by that author).

predict_cv *Predict Function for glmnet_with_cv Models*

Description

Generate predictions from the model fitted by glmnet_with_cv. This function accepts new data and returns predictions, optionally debiased if a debiasing linear model was fit.

Usage

```
predict_cv(object, newdata, debias = FALSE, ...)
```

Arguments

object	An object returned by glmnet_with_cv.
newdata	A data frame of new predictor values.
debias	Logical; if TRUE, applies the debiasing linear model stored in object$debias_fit (if available). Default is FALSE.
...	Additional arguments (not used).

Details

Predictions are computed by forming the model matrix from newdata using the stored formula and terms in the fitted model object. The coefficients used are those stored in parms. If debias=TRUE and a debias_fit linear model is available, predictions are adjusted by that model.

Value

A numeric vector of predictions.

See Also

glmnet_with_cv, SVEMnet, predict.svem_model

Examples

```
set.seed(0)
n <- 50
X1 <- runif(n)
X2 <- runif(n)
y <- 1 + 2*X1 + 3*X2 + rnorm(n)
data <- data.frame(y, X1, X2)
model_cv <- glmnet_with_cv(y ~ X1 + X2, data = data, glmnet_alpha = c(0,0.5,1))
predictions <- predict_cv(model_cv, data)
predictions_debiased <- predict_cv(model_cv, data, debias = TRUE)
```

print.svem_significance_test *Print Method for SVEM Significance Test*

Description

Prints the p-value from an object of class `svem_significance_test`.

Usage

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

Arguments

`x`	An object of class `svem_significance_test`.
`...`	Additional arguments (not used).

SVEMnet *Fit an SVEMnet Model*

Description

Wrapper for 'glmnet' (Friedman et al. 2010) to fit an ensemble of Elastic Net models using the Self-Validated Ensemble Model method (SVEM, Lemkus et al. 2021). Allows searching over multiple alpha values in the Elastic Net penalty.

Usage

```
SVEMnet(
  formula,
  data,
  nBoot = 200,
  glmnet_alpha = c(0, 0.5, 1),
  weight_scheme = c("SVEM", "FWR", "Identity"),
  objective = c("wAIC", "wSSE"),
  standardize = TRUE,
  ...
)
```

Arguments

formula	A formula specifying the model to be fitted.
data	A data frame containing the variables in the model.
nBoot	Number of bootstrap iterations (default is 200).
glmnet_alpha	Elastic Net mixing parameter(s) (default is c(0, 0.5, 1)). Can be a vector of alpha values, where alpha = 1 corresponds to Lasso and alpha = 0 corresponds to Ridge regression.
weight_scheme	Weighting scheme for SVEM (default is "SVEM"). Valid options are "SVEM", "FWR", and "Identity". "FWR" calculates the Fractional Weight Regression (Xu et al., 2020) and is included for demonstration; "SVEM" generally provides better performance."Identity" simply sets the training and validation weights to 1. Use with nBoot = 1 and objective = "wAIC" to get an elastic net fit on the training data using AIC.
objective	Objective function for selecting lambda (default is "wAIC"). Valid options are "wAIC" and "wSSE". The "w" refers to "weighted" validation.
standardize	logical. Passed to glmnet to control standardization (default is TRUE).
...	Additional arguments passed to the underlying glmnet() function.

Details

The Self-Validated Ensemble Model (SVEM, Lemkus et al., 2021) framework provides a bootstrap approach to improve predictions from various base learning models, including Elastic Net regression as implemented in 'glmnet'. SVEM is particularly suited for situations where a complex response surface is modeled with relatively few experimental runs.

In each of the 'nBoot' iterations, SVEMnet applies random exponentially distributed weights to the observations. Anti-correlated weights are used for validation.

SVEMnet allows for the Elastic Net mixing parameter ('glmnet_alpha') to be a vector, enabling the function to search over multiple 'alpha' values within each bootstrap iteration. Within each iteration, the model is fit for each specified 'alpha', and the best 'alpha' is selected based on the specified 'objective'.

objective options:

"wSSE" Weighted Sum of Squared Errors. Selects the lambda that minimizes the weighted validation error without penalizing model complexity. While this may lead to models that overfit when the number of parameters is large relative to the number of observations, SVEM mitigates overfitting (high prediction variance) by averaging over multiple bootstrap models. This is the objective function used by Lemkus et al. (2021) with weight_scheme="SVEM"

"wAIC" Weighted Akaike Information Criterion. Balances model fit with complexity by penalizing the number of parameters. It is calculated as AIC = n * log(wSSE / n) + 2 * k, where wSSE is the weighted sum of squared errors, n is the number of observations, and k is the number of parameters with nonzero coefficients. Typically used with weight_scheme="FWR" or weight_scheme="Identity"

weight_scheme options:

"SVEM" Uses anti-correlated fractional weights for training and validation sets, improving model generalization by effectively simulating multiple training-validation splits (Lemkus et al. (2021)). Published results (Lemkus et al. (2021), Karl (2024)) utilize objective="wSSE". However, unpublished simulation results suggest improved performance from using objective="wAIC" with weight_scheme="SVEM". See the SVEMnet Vignette for details.

"FWR" Fractional Weight Regression as described by Xu et al. (2020). Weights are the same for both training and validation sets. This method does not provide the self-validation benefits of SVEM but is included for comparison. Used with objective="wAIC".

"Identity" Uses weights of 1 for both training and validation. This uses the full dataset for both training and validation, effectively disabling the self-validation mechanism. Use with objective="wAIC" and nBoot=1 to fit the Elastic Net on the AIC of the training data.

A debiased fit is output (along with the standard fit). This is provided to allow the user to match the output of JMP, which returns a debiased fit whenever nBoot>=10. \https://www.jmp.com/support/help/en/18.1/?utm_source=of-selfvalidated-ensemble-models.shtml. The debiasing coefficients are always calculated by SVEMnet(), and the predict() function determines whether the raw or debiased predictions are returned via its debias argument. The default is debias=FALSE, based on performance on unpublished simulation results.

The output includes: **Model Output:** The returned object is a list of class svem_model, containing the following components:

- parms: Averaged coefficients across all bootstrap iterations.
- debias_fit: The debiasing linear model fit (if applicable). This is a linear model of the form y ~ y_pred, used to adjust the predictions and reduce bias.
- coef_matrix: Matrix of coefficients from each bootstrap iteration. Each row corresponds to a bootstrap iteration, and each column corresponds to a model coefficient.
- nBoot: Number of bootstrap iterations performed.
- glmnet_alpha: The Elastic Net mixing parameter(s) used. This is the alpha parameter from glmnet.
- best_alphas: The best alpha values selected during the fitting process for each bootstrap iteration.
- best_lambdas: The best lambda values selected during the fitting process for each bootstrap iteration.
- weight_scheme: The weighting scheme used in SVEM. Indicates whether "SVEM", "FWR", or "Identity" weights were used.
- actual_y: The response vector used in the model.
- training_X: The predictor matrix used in the model.
- y_pred: The predicted response values from the ensemble model before debiasing.
- y_pred_debiased: The debiased predicted response values (if debiasing is applied). Adjusted predictions using the debias_fit model.
- nobs: The number of observations in the dataset.
- nparm: The number of parameters (including the intercept), calculated as ncol(X) + 1.
- formula: The formula used in the model fitting.
- terms: The terms object extracted from the model frame.

Value

An object of class `svem_model`.

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024) (it was handwritten by that author).

References

Gotwalt, C., & Ramsey, P. (2018). Model Validation Strategies for Designed Experiments Using Bootstrapping Techniques With Applications to Biopharmaceuticals. *JMP Discovery Conference*. https://community.jmp.com/t5/Discovery-Summit-2018/Model-Validation-Strategies-for-Designed-Experi ta-p/73730

Karl, A. T. (2024). A randomized permutation whole-model test heuristic for Self-Validated Ensemble Models (SVEM). *Chemometrics and Intelligent Laboratory Systems*, *249*, 105122. doi:10.1016/j.chemolab.2024.105122

Karl, A., Wisnowski, J., & Rushing, H. (2022). JMP Pro 17 Remedies for Practical Struggles with Mixture Experiments. JMP Discovery Conference. doi:10.13140/RG.2.2.34598.40003/1

Lemkus, T., Gotwalt, C., Ramsey, P., & Weese, M. L. (2021). Self-Validated Ensemble Models for Design of Experiments. *Chemometrics and Intelligent Laboratory Systems*, 219, 104439. doi:10.1016/j.chemolab.2021.104439

Xu, L., Gotwalt, C., Hong, Y., King, C. B., & Meeker, W. Q. (2020). Applications of the Fractional-Random-Weight Bootstrap. *The American Statistician*, 74(4), 345–358. doi:10.1080/00031305.2020.1731599

Ramsey, P., Gaudard, M., & Levin, W. (2021). Accelerating Innovation with Space Filling Mixture Designs, Neural Networks and SVEM. *JMP Discovery Conference*. https://community.jmp.com/t5/Abstracts/Accelerating-Innovation-with-Space-Filling-Mixture-Designs/ev-p/756841

Ramsey, P., & Gotwalt, C. (2018). Model Validation Strategies for Designed Experiments Using Bootstrapping Techniques With Applications to Biopharmaceuticals. *JMP Discovery Conference - Europe*. https://community.jmp.com/t5/Discovery-Summit-Europe-2018/Model-Validation-Strategies-for-De ta-p/51286

Ramsey, P., Levin, W., Lemkus, T., & Gotwalt, C. (2021). SVEM: A Paradigm Shift in Design and Analysis of Experiments. *JMP Discovery Conference - Europe*. https://community.jmp.com/t5/Abstracts/SVEM-A-Paradigm-Shift-in-Design-and-Analysis-of-Experiments-2021/ev-p/756634

Ramsey, P., & McNeill, P. (2023). CMC, SVEM, Neural Networks, DOE, and Complexity: It's All About Prediction. *JMP Discovery Conference*.

Friedman, J. H., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–22.

Examples

```
# Simulate data
set.seed(0)
```

```
n <- 21
X1 <- runif(n)
X2 <- runif(n)
X3 <- runif(n)
y <- 1 + 2*X1 + 3*X2 + X1*X2 + X1^2  + rnorm(n)
data <- data.frame(y, X1, X2, X3)

# Fit the SVEMnet model with a formula
model <- SVEMnet(
  y ~ (X1 + X2 + X3)^2 + I(X1^2) + I(X2^2) + I(X3^2),
  glmnet_alpha = c(1),
  data = data
)
coef(model)
plot(model)
predict(model,data)
```

`svem_significance_test` *SVEM Significance Test*

Description

Performs a whole-model significance test using the SVEM framework, handling both continuous and categorical predictors.

Usage

```
svem_significance_test(
  formula,
  data,
  nPoint = 2000,
  nSVEM = 5,
  nPerm = 125,
  percent = 85,
  nBoot = 200,
  glmnet_alpha = c(1),
  weight_scheme = c("SVEM"),
  objective = c("wAIC", "wSSE"),
  verbose = TRUE,
  ...
)
```

Arguments

`formula`	A formula specifying the model to be tested.
`data`	A data frame containing the variables in the model.

nPoint	The number of random points to generate in the factor space (default: 2000).
nSVEM	The number of SVEM models to fit to the original data (default: 5).
nPerm	The number of SVEM models to fit to permuted data for reference distribution (default: 125).
percent	The percentage of variance to capture in the SVD (default: 85).
nBoot	The number of bootstrap iterations within SVEM (default: 200).
glmnet_alpha	The alpha parameter(s) for glmnet (default: c(1)).
weight_scheme	The weight scheme to use in SVEM (default: "SVEM").
objective	Character; the objective function to use in SVEMnet. Options are "wAIC" or "wSSE" (default: "wAIC").
verbose	Logical; if TRUE, displays progress messages (default: TRUE).
...	Additional arguments passed to the underlying SVEMnet() and then glmnet() functions.

Details

The 'svem_significance_test' function implements a whole-model test designed to gauge the significance of a fitted SVEM model compared to the null hypothesis of a constant response surface. This method helps identify responses that have relatively stronger or weaker relationships with study factors.

The test constructs standardized predictions by centering the SVEM predictions (obtained from SVEMnet()) by the response mean and scaling by the ensemble standard deviation. A reference distribution is created by fitting the SVEM model to multiple randomized permutations of the response vector. The Mahalanobis distances of the original and permuted models are calculated using a reduced-rank singular value decomposition.

The R code to perform this test (using matrices of nSVEM and nPerm predictions) is taken from the supplementary material of Karl (2024).

This function assumes that there are no restrictions among the factors (e.g. mixture factors). The method will work with restrictions, but the code would need to be changed to ensure the nPoint points respect the factor restriction(s). For example, rdirichlet() could be used for the mixture factors.

The SVEMnet parameter debias is hard coded to FALSE for this test. Unpublished simulation work suggests that setting debias=TRUE reduces the power of the test (without affecting the Type I error rate).

Value

A list containing the test results.

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024) (it was handwritten by that author).

References

Karl, A. T. (2024). A randomized permutation whole-model test heuristic for Self-Validated Ensemble Models (SVEM). *Chemometrics and Intelligent Laboratory Systems*, *249*, 105122. doi:10.1016/j.chemolab.2024.105122

Examples

```
# Simulate data
set.seed(1)
n <- 30
X1 <- runif(n)
X2 <- runif(n)
X3 <- runif(n)
y <- 1 + X1 +  X2 + X1 * X2 + X1^2 + rnorm(n)
data <- data.frame(y, X1, X2, X3)

# Perform the SVEM significance test
test_result <- svem_significance_test(
  y ~ (X1 + X2 + X3)^2 + I(X1^2) + I(X2^2) + I(X3^2),
  data = data,
  nPoint = 2000,
  nSVEM = 7,
  nPerm = 150,
  nBoot = 200

)

# View the p-value
print(test_result)


test_result2 <- svem_significance_test(
  y ~ (X1 + X2 )^2 + I(X1^2) + I(X2^2),
  data = data,
  nPoint = 2000,
  nSVEM = 7,
  nPerm = 150,
  nBoot = 200
)

# View the p-value
print(test_result2)


# Plot the Mahalanobis distances
plot(test_result,test_result2)
```

svem_significance_test_parallel *SVEM Significance Test (Parallel Version)*

Description

Performs a whole-model significance test using the SVEM framework, handling both continuous and categorical predictors, with parallel computation.

Usage

```
svem_significance_test_parallel(
  formula,
  data,
  nPoint = 2000,
  nSVEM = 7,
  nPerm = 200,
  percent = 85,
  nBoot = 200,
  glmnet_alpha = c(1),
  weight_scheme = c("SVEM"),
  objective = c("wAIC", "wSSE"),
  verbose = TRUE,
  nCore = parallel::detectCores(),
  seed = NULL,
  ...
)
```

Arguments

formula	A formula specifying the model to be tested.
data	A data frame containing the variables in the model.
nPoint	The number of random points to generate in the factor space (default: 2000).
nSVEM	The number of SVEM models to fit to the original data (default: 7).
nPerm	The number of SVEM models to fit to permuted data for reference distribution (default: 200).
percent	The percentage of variance to capture in the SVD (default: 85).
nBoot	The number of bootstrap iterations within SVEM (default: 200).
glmnet_alpha	The alpha parameter(s) for glmnet (default: c(1)).
weight_scheme	The weight scheme to use in SVEM (default: "SVEM").
objective	Character; the objective function to use in SVEMnet. Options are "wAIC" or "wSSE" (default: "wAIC").
verbose	Logical; if TRUE, displays progress messages (default: TRUE).

nCore	The number of CPU cores to use for parallel processing (default: all available cores).
seed	An integer seed for random number generation (default: NULL).
...	Additional arguments passed to the underlying SVEMnet() and then glmnet() functions.

Details

The 'svem_significance_test_parallel' function implements a whole-model test designed to gauge the significance of a fitted SVEM model compared to the null hypothesis of a constant response surface, with parallel computation. This method helps identify responses that have relatively stronger or weaker relationships with study factors.

The test constructs standardized predictions by centering the SVEM predictions (obtained from SVEMnet()) by the response mean and scaling by the ensemble standard deviation. A reference distribution is created by fitting the SVEM model to multiple randomized permutations of the response vector. The Mahalanobis distances of the original and permuted models are calculated using a reduced-rank singular value decomposition.

The R code to perform this test (using matrices of nSVEM and nPerm predictions) is taken from the supplementary material of Karl (2024).

This function assumes that there are no restrictions among the factors (e.g. mixture factors). The method will work with restrictions, but the code would need to be changed to ensure the nPoint points respect the factor restriction(s). For example, rdirichlet() could be used for the mixture factors.

The SVEMnet parameter debias is hard coded to FALSE for this test. Unpublished simulation work suggests that setting debias=TRUE reduces the power of the test (without affecting the Type I error rate).

Value

A list containing the test results.

Acknowledgments

Development of this package was assisted by GPT o1-preview, which helped in constructing the structure of some of the code and the roxygen documentation. The code for the significance test is taken from the supplementary material of Karl (2024).

References

Karl, A. T. (2024). A randomized permutation whole-model test heuristic for Self-Validated Ensemble Models (SVEM). *Chemometrics and Intelligent Laboratory Systems*, *249*, 105122. doi:10.1016/j.chemolab.2024.105122

Examples

```
# Simulate data
set.seed(0)
n <- 30
```

```
X1 <- runif(n)
X2 <- runif(n)
X3 <- runif(n)
y <- 1 + X1 +  X2 + X1 * X2 + X1^2 + rnorm(n)
data <- data.frame(y, X1, X2, X3)

#CRAN requires a max of nCore=2 for example. Recommend using default nCore to use entire CPU.

# Perform the SVEM significance test
test_result <- svem_significance_test_parallel(
  y ~ (X1 + X2 + X3)^2 + I(X1^2) + I(X2^2) + I(X3^2),
  data = data,
  nPoint = 2000,
  nSVEM = 9,
  nPerm = 250,
  nBoot = 200,
  nCore = 2
)

# View the p-value
print(test_result)


test_result2 <- svem_significance_test_parallel(
  y ~ (X1 + X2 )^2 + I(X1^2) + I(X2^2),
  data = data,
  nPoint = 2000,
  nSVEM = 9,
  nPerm = 250,
  nBoot = 200,
  nCore = 2
)

# View the p-value
print(test_result2)


# Plot the Mahalanobis distances
plot(test_result,test_result2)
```

Index