Package 'SVEMnet'

August 23, 2025

Type Package

Title Self-Validated Ensemble Models with Elastic Net Regression

Version 1.5.3

Date 2025-08-21

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] (ORCID: <https://orcid.org/0000-0002-5933-8706>)

Repository CRAN

Date/Publication 2025-08-22 23:10:02 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/Abstracts/Model-Validation-Strategies-for-Designed-Experiments-Using/ev-p/849873/redirect_from_archived_page/true

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/Abstracts/Model-Validation-Strategies-for-Designed-Experiments-ev-p/849647/redirect_from_archived_page/true

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(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(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,
  agg = c("parms", "mean"),
  ...
)
```

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).
agg	Aggregation method for ensemble predictions. One of "parms" (default) or "mean". "parms" uses the aggregated coefficients stored in object$parms (or parms_debiased if debias=TRUE). "mean" averages predictions from individual bootstrap members equally.
...	Additional arguments (currently unused).

Details

A debiased fit is available (along with the standard fit). This is provided to allow the user to match the output of JMP. The debiasing coefficients are always calculated by SVEMnet(), and the predict() function determines whether the raw or debiased predictions are returned via the debias argument (default FALSE). When se.fit=TRUE and debias=TRUE, the reported SE is the bootstrap spread scaled by $|b|$ from the calibration $y \sim y_{pred}$.

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 Method for glmnet_with_cv Objects*

Description

Generates predictions from a fitted object returned by `glmnet_with_cv()`.

Usage

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

Arguments

`object`	A list returned by `glmnet_with_cv()`.
`newdata`	A data frame of new predictor values.
`debias`	Logical; if `TRUE` and a debiasing fit is available, apply it (default `FALSE`).
`strict`	Logical; if `TRUE`, require exact column-name match with training design (default `FALSE`).
`...`	Additional arguments (currently unused).

Details

Columns are aligned by name. With `strict=TRUE`, a mismatch errors.

Value

A numeric vector of predictions.

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 = 300,
  glmnet_alpha = c(1),
  weight_scheme = c("SVEM", "FWR", "Identity"),
  objective = c("wAIC", "wBIC", "wGIC", "wSSE"),
  gamma = 2,
  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 300).
glmnet_alpha	Elastic Net mixing parameter(s) (default is c(1)). Can be a vector of alpha values, where alpha = 1 is Lasso and alpha = 0 is Ridge.
weight_scheme	Weighting scheme for SVEM (default "SVEM"). One of "SVEM", "FWR", or "Identity".
objective	Objective used to pick lambda on each bootstrap path (default "wAIC"). One of "wAIC", "wBIC", "wGIC", or "wSSE".
gamma	Penalty weight used only when objective="wGIC" (numeric, default 2). Setting gamma = 2 reproduces wAIC.
standardize	Logical; passed to glmnet (default TRUE).
...	Additional args to glmnet().

Details

The Self-Validated Ensemble Model (SVEM, Lemkus et al., 2021) framework provides a bootstrap approach to improve predictions from base learners, including Elastic Net regression as implemented in glmnet. In each of the nBoot iterations, SVEMnet applies random exponentially distributed weights to the observations; anti-correlated weights are used for validation when weight_scheme="SVEM".

SVEMnet allows glmnet_alpha to be a vector, enabling a search over multiple Elastic Net mixing parameters within each bootstrap. The objective controls how the validation criterion balances fit and complexity:

"wSSE" Weighted Sum of Squared Errors: uses the weighted validation SSE directly.

"wAIC" Weighted AIC (Gaussian): AIC = n * log(SSE_w / n) + 2 * k, where n = sum(w_valid) (after normalization) and k counts parameters including the intercept. Candidates require k < n.

"wBIC" Weighted BIC-like criterion: n * log(SSE_w / n) + log(n_eff) * k, with n_eff = (sum(w_valid)^2) / sum(w_valid^2) (Kish). For stability, n_eff is clipped to [5, n]. Candidates require k < n and k < n_eff - 1.

"wGIC" Weighted Generalized Information Criterion: n * log(SSE_w / n) + gamma * k. Here gamma is a fixed nonnegative number. For robustness near the boundary, candidates require k < n and k < n_eff - 1.

Note on BIC: In reweighted validation, information content varies with weight heterogeneity; using log(n_eff) adapts the penalty to that effective size. With uniform weights (Identity), n_eff = n and you recover standard BIC.

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. The debiasing coefficients are always calculated by SVEMnet(), and the predict() method determines whether the raw or debiased predictions are returned via its debias argument. The default is debias = FALSE, based on performance on unpublished simulations.

The returned object includes averaged coefficients (`parms`), debiased coefficients (`parms_debiased`), the calibration fit (`debias_fit`), per-bootstrap coefficients, chosen alphas and lambdas, the chosen `objective` (and `gamma` if applicable), and a compact `diagnostics` list (median/IQR of selected model size and alpha frequencies).

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/Abstracts/Model-Validation-Strategies-for-Designed-Experiments-Using/ev-p/849873/redirect_from_archived_page/true

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/Abstracts/Model-Validation-Strategies-for-Designed-Experiments-ev-p/849647/redirect_from_archived_page/true

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)

# Example: BIC-like penalty
# model_bic <- SVEMnet(y ~ X1 + X2 + X3, data = data, objective = "wBIC")
# Example: GIC with custom gamma
# model_gic <- SVEMnet(y ~ X1 + X2 + X3, data = data, objective = "wGIC", gamma = 4)
```

---

svem_random_table *Generate a Random Prediction Table for a Fitted SVEMnet Model*

Description

This utility function generates a random sample of points from the predictor space and computes the corresponding predicted responses from a fitted SVEMnet model. It can be used to explore the fitted response surface in a way analogous to JMP's "Output Random Table" feature. The function recognizes mixture factor groups and draws Dirichlet-distributed compositions within the specified bounds so that mixture variables sum to a user-supplied total. Continuous non-mixture variables are sampled uniformly across their observed ranges using a maximin Latin hypercube design, and categorical variables are sampled from their observed levels. No random noise is added to the predicted responses.

Usage

```
svem_random_table(
  formula,
  data,
  n = 1000,
  mixture_groups = NULL,
  nBoot = 200,
```

```
  glmnet_alpha = c(1),
  weight_scheme = c("SVEM"),
  objective = c("wAIC", "wSSE"),
  debias = FALSE,
  ...
)
```

Arguments

formula	A formula specifying the fitted model. This should be the same formula used when fitting the SVEMnet model.
data	A data frame containing the variables in the model.
n	Number of random points to generate (default: 1000).
mixture_groups	Optional list describing mixture factor groups. Each element should be a list with components ‘vars‘ (character vector of mixture variable names), ‘lower‘ (numeric vector of lower bounds), ‘upper‘ (numeric vector of upper bounds) and ‘total‘ (scalar sum). See ‘svem_significance_test_with_mixture()‘ for details. Defaults to ‘NULL‘ (no mixtures).
nBoot	Number of bootstrap iterations to use when fitting the SVEMnet model (default: 200).
glmnet_alpha	Elastic net mixing parameter(s) passed to ‘SVEMnet‘ (default: ‘c(1)‘).
weight_scheme	Weighting scheme for SVEM (default: "SVEM").
objective	Objective function for SVEM ("wAIC" or "wSSE"; default: "wAIC").
debias	Logical; if ‘TRUE‘, the debiasing coefficients of the fitted model are applied when predicting (default: ‘FALSE‘).
...	Additional arguments passed to ‘SVEMnet()‘ and then to ‘glmnet()‘.

Details

This function first fits an SVEMnet model using the supplied parameters. It then generates a random grid of points in the predictor space, honouring mixture constraints if ‘mixture_groups‘ is provided. Predictions are computed from the fitted model on these points. No random noise is added; the predictions come directly from the model. If you wish to explore the uncertainty of predictions, consider adding noise separately or using the standard error output from ‘predict.svem_model()‘.

Value

A data frame containing the sampled predictor values and the corresponding predicted responses. The response column is named according to the left-hand side of ‘formula‘.

See Also

‘SVEMnet‘, ‘predict.svem_model‘, ‘svem_significance_test_with_mixture‘.

Examples

```
set.seed(42)
n <- 40

# Helper to generate training data mixtures with bounds
sample_trunc_dirichlet <- function(n, lower, upper, total) {
  k <- length(lower)
  min_sum <- sum(lower); max_sum <- sum(upper)
  stopifnot(total >= min_sum, total <= max_sum)
  avail <- total - min_sum
  out <- matrix(NA_real_, n, k)
  i <- 1
  while (i <= n) {
    g <- rgamma(k, 1, 1)
    w <- g / sum(g)
    x <- lower + avail * w
    if (all(x <= upper + 1e-12)) {
      out[i, ] <- x; i <- i + 1
    }
  }
  out
}

# Three mixture factors (A, B, C) with distinct bounds; sum to total = 1
lower <- c(0.10, 0.20, 0.05)
upper <- c(0.60, 0.70, 0.50)
total <- 1.0
ABC   <- sample_trunc_dirichlet(n, lower, upper, total)
A <- ABC[, 1]; B <- ABC[, 2]; C <- ABC[, 3]

# Additional predictors
X <- runif(n)
F <- factor(sample(letters[1:3], n, replace = TRUE))

# Response
y <- 1 + 2*A + 3*B + 1.5*C + 0.5*X +
     ifelse(F == "a", 0, ifelse(F == "b", 1, -1)) +
     rnorm(n, sd = 0.3)

dat <- data.frame(y = y, A = A, B = B, C = C, X = X, F = F)

# Mixture specification for the random table generator
mix_spec <- list(
  list(
    vars  = c("A", "B", "C"),
    lower = c(0.10, 0.20, 0.05),
    upper = c(0.60, 0.70, 0.50),
    total = 1.0
  )
)

# Fit SVEMnet and generate 50 random points
```

```
rand_tab <- svem_random_table(
  y ~ A + B + C + X + F,
  data           = dat,
  n              = 50,
  mixture_groups = mix_spec,
  nBoot          = 50,
  glmnet_alpha   = c(1),
  weight_scheme  = "SVEM",
  objective      = "wAIC",
  debias         = FALSE
)

# Check mixture validity in the generated table
stopifnot(all(abs((rand_tab$A + rand_tab$B + rand_tab$C) - 1) < 1e-8))
summary(rand_tab[c("A","B","C")])
head(rand_tab)
```

`svem_significance_test`

SVEM Significance Test with Mixture Support

Description

Performs a whole-model significance test using the SVEM framework and allows the user to specify mixture factor groups. Mixture factors are sets of continuous variables that are constrained to sum to a constant (the mixture total) and have optional lower and upper bounds. When mixture groups are supplied, the grid of evaluation points is generated by sampling Dirichlet variates over the mixture simplex rather than by independently sampling each continuous predictor. Non-mixture continuous predictors are sampled via a maximin Latin hypercube over their observed ranges, and categorical predictors are sampled from their observed levels. The remainder of the algorithm follows 'svem_significance_test()', computing standardized predictions on the grid, refitting SVEM on permutations of the response, and calculating a Mahalanobis distance for the original and permutation fits.

Usage

```
svem_significance_test(
  formula,
  data,
  mixture_groups = NULL,
  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.
mixture_groups	Optional list describing one or more mixture factor groups. Each element of the list should be a list with components 'vars' (character vector of column names), 'lower' (numeric vector of lower bounds of the same length as 'vars'), 'upper' (numeric vector of upper bounds of the same length), and 'total' (scalar specifying the sum of the mixture variables). All mixture variables must be included in 'vars', and no variable can appear in more than one mixture group. Defaults to 'NULL' (no mixtures).
nPoint	Number of random points in the factor space (default: 2000).
nSVEM	Number of SVEM fits on the original data (default: 5).
nPerm	Number of SVEM fits on permuted responses for the reference distribution (default: 125).
percent	Percentage of variance to capture in the SVD (default: 85).
nBoot	Number of bootstrap iterations within each SVEM fit (default: 200).
glmnet_alpha	The alpha parameter(s) for glmnet (default: 'c(1)').
weight_scheme	Weighting scheme for SVEM (default: "SVEM").
objective	Objective function for SVEM ("wAIC" or "wSSE", default: "wAIC").
verbose	Logical; if 'TRUE', displays progress messages (default: 'TRUE').
...	Additional arguments passed to 'SVEMnet()' and then to 'glmnet()'.

Details

This function extends 'svem_significance_test()' by allowing the user to specify mixture factor groups. In a mixture group, the specified variables are jointly sampled from a Dirichlet distribution so that their values sum to the specified 'total'. Lower and upper bounds can be supplied to shift and scale the mixture simplex. Feasibility is checked ('sum(lower) <= total <= sum(upper)'), and samples are generated as 'lower + (total - sum(lower)) * w' for Dirichlet weights 'w', with rejection of any draws violating the upper bounds. This guarantees the correct total while respecting all bounds.

If no mixture groups are supplied, this function behaves identically to 'svem_significance_test()'.

Value

A list of class 'svem_significance_test' containing the test results.

See Also

'svem_significance_test()'

Examples

```
# Construct a small data set with a three-component mixture (A, B, C)
# Each has distinct lower/upper bounds and they sum to 1
set.seed(123)
n <- 30

# Helper used only for generating training data in this example
sample_trunc_dirichlet <- function(n, lower, upper, total) {
  k <- length(lower)
  min_sum <- sum(lower); max_sum <- sum(upper)
  stopifnot(total >= min_sum, total <= max_sum)
  avail <- total - min_sum
  out <- matrix(NA_real_, n, k)
  i <- 1L
  while (i <= n) {
    g <- rgamma(k, 1, 1)
    w <- g / sum(g)
    x <- lower + avail * w
    if (all(x <= upper + 1e-12)) {
      out[i, ] <- x
      i <- i + 1L
    }
  }
  out
}

# Three mixture components with distinct bounds; sum to 1
lower <- c(0.10, 0.20, 0.05)  # for A, B, C
upper <- c(0.60, 0.70, 0.50)
total <- 1.0
ABC   <- sample_trunc_dirichlet(n, lower, upper, total)
A <- ABC[, 1]; B <- ABC[, 2]; C <- ABC[, 3]

# Additional predictors
X <- runif(n)
F <- factor(sample(c("red", "blue"), n, replace = TRUE))

# Response
y <- 2 + 3*A + 1.5*B + 1.2*C + 0.5*X + 1*(F == "red") + rnorm(n, sd = 0.3)
dat <- data.frame(y = y, A = A, B = B, C = C, X = X, F = F)

# Specify the mixture group for A, B, C
mix_spec <- list(
  list(
    vars  = c("A", "B", "C"),
    lower = c(0.10, 0.20, 0.05),
    upper = c(0.60, 0.70, 0.50),
    total = 1.0
  )
)

# Run the whole-model significance test on this mixture model
```

```
test_res <- svem_significance_test(
  y ~ A + B + C + X + F,
  data           = dat,
  mixture_groups = mix_spec,
  nPoint         = 200,
  nSVEM          = 3,
  nPerm          = 50,
  nBoot          = 100,
  glmnet_alpha   = c(1),
  weight_scheme  = "SVEM",
  objective      = "wAIC",
  verbose        = FALSE
)

print(test_res)
plot(test_res)
```

`svem_significance_test_parallel` *SVEM Significance Test with Mixture Support (Parallel Version)*

Description

Whole-model significance test using SVEM with support for mixture factor groups, parallelizing the SVEM fits for originals and permutations.

Usage

```
svem_significance_test_parallel(
  formula,
  data,
  mixture_groups = NULL,
  nPoint = 2000,
  nSVEM = 5,
  nPerm = 125,
  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.
mixture_groups	Optional list describing one or more mixture factor groups. Each element of the list should be a list with components ‘vars‘ (character vector of column names), ‘lower‘ (numeric vector of lower bounds of the same length as ‘vars‘), ‘upper‘ (numeric vector of upper bounds of the same length), and ‘total‘ (scalar specifying the sum of the mixture variables). All mixture variables must be included in ‘vars‘, and no variable can appear in more than one mixture group. Defaults to ‘NULL‘ (no mixtures).
nPoint	Number of random points in the factor space (default: 2000).
nSVEM	Number of SVEM fits on the original data (default: 5).
nPerm	Number of SVEM fits on permuted responses for the reference distribution (default: 125).
percent	Percentage of variance to capture in the SVD (default: 85).
nBoot	Number of bootstrap iterations within each SVEM fit (default: 200).
glmnet_alpha	The alpha parameter(s) for glmnet (default: ‘c(1)‘).
weight_scheme	Weighting scheme for SVEM (default: "SVEM").
objective	Objective function for SVEM ("wAIC" or "wSSE", default: "wAIC").
verbose	Logical; if ‘TRUE‘, displays progress messages (default: ‘TRUE‘).
nCore	Number of CPU cores for parallel processing (default: all available cores).
seed	Optional integer seed for reproducible parallel RNG (default: NULL).
...	Additional arguments passed to ‘SVEMnet()‘ and then to ‘glmnet()‘.

Details

Identical to `svem_significance_test()` but runs the expensive SVEM refits in parallel using `foreach` + `doParallel`. Random draws (including permutations) use `RNGkind("L'Ecuyer-CMRG")` for parallel-suitable streams.

Value

A list of class ‘svem_significance_test‘ containing the test results.

See Also

svem_significance_test svem_significance_test_parallel

Index