Package 'SVEMnet'

September 9, 2025

Type Package

Title Self-Validated Ensemble Models with Lasso and Relaxed-Elastic Net Regression

Version 2.1.3

Date 2025-09-08

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 2025-09-09 06:50:30 UTC

Contents

SVEMnet-package . . . 2
coef.svem_model . . . 3
glmnet_with_cv . . . 4
plot.svem_model . . . 6
plot_svem_significance_tests . . . 7
predict.svem_model . . . 7
predict_cv . . . 9
print.svem_significance_test . . . 10
SVEMnet . . . 10
svem_random_table_from_model . . . 14
svem_significance_test . . . 16
svem_significance_test_parallel . . . 18

Index **22**

`SVEMnet-package` *SVEMnet: Self-Validated Ensemble Models with Relaxed Lasso and 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.

`predict.svem_model` Predict method for SVEM models.

`svem_significance_test` Whole-model significance test.

`svem_significance_test_parallel` Parallel whole-model test.

`plot.svem_model` Plot actual vs predicted for a model.

`coef.svem_model` Coefficient nonzero percentages (bootstrap).

`plot_svem_significance_tests` Plotting helper for multiple tests.

`glmnet_with_cv` Wrapper around `cv.glmnet()` with repeats.

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` *Coefficient Nonzero Percentages from an SVEM Model*

Description

Calculates the percentage of bootstrap iterations in which each coefficient (excluding the intercept) is nonzero, using a small tolerance.

Usage

```
## S3 method for class 'svem_model'
coef(object, tol = 1e-07, ...)
```

Arguments

object	An object of class svem_model.
tol	Numeric tolerance for "nonzero" (default 1e-7).
...	Unused.

Value

Invisibly returns a data frame with variables and percentages.

---

glmnet_with_cv *Fit a glmnet Model with Cross-Validation*

Description

Repeated K-fold CV over a per-alpha lambda path, with a proper 1-SE rule across repeats. Preserves fields expected by predict_cv(). Optionally uses glmnet's built-in relaxed elastic net for both the warm-start path and each CV fit. When relaxed=TRUE, the final coefficients are taken from a cv.glmnet() object at the chosen lambda so that the returned model reflects the relaxed solution (including its gamma).

Usage

```
glmnet_with_cv(
  formula,
  data,
  glmnet_alpha = c(0, 0.25, 0.5, 0.75, 1),
  standardize = TRUE,
  nfolds = 10,
  repeats = 5,
  choose_rule = c("min", "1se"),
  seed = NULL,
  exclude = NULL,
  relaxed = FALSE,
  relax_gamma = NULL,
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
glmnet_alpha	Numeric vector of alphas, default c(0,0.25,0.5,0.75, 1).
standardize	Logical passed to glmnet (default TRUE).
nfolds	CV folds (default 10), internally constrained so >= ~3 obs/fold.
repeats	Number of independent CV repeats (default 5).

choose_rule	"1se" or "min" (default). In simulations "1se" lead to increased RMSE on hold-out data when simulating small mixture designs.
seed	Optional integer seed for reproducible fold IDs.
exclude	Optional vector OR function for glmnet's exclude=. If a function, cv.glmnet applies it inside each training fold (glmnet >= 4.1-2).
relaxed	Logical; if TRUE, call glmnet/cv.glmnet with relax=TRUE (default FALSE).
relax_gamma	Optional numeric vector passed as gamma= to glmnet/cv.glmnet when relaxed=TRUE. If NULL, glmnet uses its internal default gamma path.
...	Args forwarded to both cv.glmnet() and glmnet(), e.g. family, weights, parallel, type.measure, intercept, maxit, lower.limits, upper.limits, penalty.factor, offset, standardize.response, keep, etc.

Details

To avoid duplicate-argument errors, arguments like x, y, alpha, lambda, foldid, and exclude are passed explicitly and removed from the dots before calling glmnet::cv.glmnet().

Value

A list with elements:

- parms Named numeric vector of coefficients (including "(Intercept)").
- glmnet_alpha Numeric vector of alphas searched.
- best_alpha Numeric; winning alpha.
- best_lambda Numeric; winning lambda.
- y_pred In-sample predictions from the returned coefficients.
- debias_fit Optional lm(y ~ y_pred) for Gaussian family.
- y_pred_debiased If debias_fit exists, its fitted values.
- cv_summary Per-alpha data frames with lambda, mean_cvm, sd_cvm, se_combined, n_repeats, idx_min, idx_1se.
- formula Original formula.
- terms Cleaned training terms (environment set to baseenv()).
- training_X Training design matrix without intercept.
- actual_y Training response vector.
- xlevels Factor levels seen during training (for safe predict).
- contrasts Contrasts used during training (for safe predict).
- schema list(feature_names, terms_str, xlevels, contrasts, terms_hash) for deterministic predict.
- note Character vector of notes (e.g., dropped rows, relaxed-coef source).
- meta List: nfolds, repeats, rule, family, relaxed, relax_cv_fallbacks, cv_object (if keep=TRUE for the final fit).

Examples

```
set.seed(123)
n <- 100; p <- 10
X <- matrix(rnorm(n * p), n, p)
beta <- c(1, -1, rep(0, p - 2))
y <- as.numeric(X %*% beta + rnorm(n))
df_ex <- data.frame(y = y, X)
colnames(df_ex) <- c("y", paste0("x", 1:p))

# Default: 1SE rule, repeats=1, non-relaxed
fit_1se <- glmnet_with_cv(y ~ ., df_ex, glmnet_alpha = c(0.5, 1),
                          nfolds = 5, repeats = 1, seed = 42)
str(fit_1se$parms)

# v1-like behavior: choose_rule="min"
fit_min <- glmnet_with_cv(y ~ ., df_ex, glmnet_alpha = 1,
                          nfolds = 5, repeats = 1, choose_rule = "min", seed = 42)

# Relaxed path with gamma search
fit_relax <- glmnet_with_cv(y ~ ., df_ex, glmnet_alpha = 1,
                            nfolds = 5, repeats = 1, relaxed = TRUE, seed = 42)
```

---

plot.svem_model *Plot Method for SVEM Models*

Description

Plots actual versus predicted values for an svem_model.

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.
...	Additional aesthetics passed to geom_point().

Value

A ggplot object.

plot_svem_significance_tests *Plot SVEM Significance Test Results for Multiple Responses*

Description

Plots the Mahalanobis-like distances for original and permuted data from one or more SVEM significance test results.

Usage

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

Arguments

x	An object of class svem_significance_test.
...	Optional additional svem_significance_test objects to include.
labels	Optional character vector of labels for the responses. If not provided, the function uses the response names and ensures uniqueness.

Details

If additional svem_significance_test objects are provided via ..., they will be combined into a single plot alongside x.

Value

A ggplot object showing the distributions of distances.

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 (created by SVEMnet()).
newdata	A data frame of new predictor values.
debias	Logical; default is FALSE. If TRUE, apply the linear calibration fit (y ~ y_pred) learned during training when available.
se.fit	Logical; if TRUE, returns standard errors (default 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 and optionally applies the debias calibration.
...	Additional arguments (currently unused).

Details

The function uses the training terms, factor levels (xlevels), and contrasts saved by SVEMnet(). The terms object environment is set to baseenv() to avoid unexpected lookup of objects in the original environment.

Column handling is strict but robust:

- We require that the set of columns produced by `model.matrix` on `newdata` matches the set used in training.
- If `model.matrix` drops some training columns (e.g., a factor collapses to a single level in `newdata`), we add those missing columns as zeros so coefficients align.
- We then reorder columns to the exact training order before multiplying.

Rows in `newdata` that contain unseen factor levels will yield NA predictions (and NA standard errors if `se.fit=TRUE`); a warning is issued indicating how many rows were affected.

When `agg="mean"`, the predictive SE is the bootstrap spread (row-wise sd of member predictions). If `debias=TRUE` and the calibration slope is available and finite, SEs are scaled by abs(slope).

Value

A numeric vector of predictions, or a list with components `fit` and `se.fit` when `se.fit = TRUE`.

Examples

```
set.seed(1)
n  <- 40
X1 <- rnorm(n); X2 <- rnorm(n); X3 <- rnorm(n)
y  <- 1 + 0.8*X1 - 0.5*X2 + 0.2*X3 + rnorm(n, 0, 0.3)
dat <- data.frame(y, X1, X2, X3)
fit <- SVEMnet(y ~ (X1 + X2 + X3)^2, dat, nBoot = 30, relaxed = TRUE)
pred <- predict(fit, dat)
head(pred)

# With SEs and debias
out <- predict(fit, dat, debias = TRUE, se.fit = TRUE, agg = "mean")
str(out)
```

predict_cv *Predict for svem_cv objects (and convenience wrapper)*

Description

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

Usage

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

## S3 method for class 'svem_cv'
predict(object, newdata, debias = FALSE, strict = FALSE, ...)
```

Arguments

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

Details

The design matrix for `newdata` is rebuilt using the training `terms` (with environment set to `baseenv()`), along with the saved factor `xlevels` and `contrasts` (stored in `object$schema`). Columns are aligned robustly to the training order:

- Any training columns that `model.matrix()` drops for `newdata` (e.g., a factor collapses to a single level) are added back as zero columns.
- Columns are reordered to exactly match the training order.
- Rows with unseen factor levels are warned about and return `NA`.

If `debias = TRUE` and a calibration fit `lm(y ~ y_pred)` exists with a finite slope, predictions are transformed by `a + b * pred`.

Value

A numeric vector of predictions.

Examples

```
set.seed(1)
n <- 50; p <- 5
X <- matrix(rnorm(n * p), n, p)
y <- X[,1] - 0.5 * X[,2] + rnorm(n)
df_ex <- data.frame(y = as.numeric(y), X)
colnames(df_ex) <- c("y", paste0("x", 1:p))
fit <- glmnet_with_cv(y ~ ., df_ex, glmnet_alpha = 1, nfolds = 5, repeats = 2, seed = 9)
preds_raw <- predict_cv(fit, df_ex)
preds_db  <- predict_cv(fit, df_ex, debias = TRUE)
cor(preds_raw, df_ex$y)
```

---

print.svem_significance_test

Print Method for SVEM Significance Test

Description

Prints the median 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 (unused).

SVEMnet

Fit an SVEMnet Model (with optional relaxed elastic net)

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), with an option to use glmnet's built-in relaxed elastic net (Meinshausen 2007). Supports searching over multiple alpha values in the Elastic Net penalty.

Usage

```
SVEMnet(
  formula,
  data,
  nBoot = 200,
  glmnet_alpha = c(0.25, 0.5, 0.75, 1),
  weight_scheme = c("SVEM", "FWR", "Identity"),
  objective = c("auto", "wAIC", "wBIC", "wGIC", "wSSE"),
  auto_ratio_cutoff = 1.3,
  gamma = 2,
  relaxed = 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 200).
glmnet_alpha	Elastic Net mixing parameter(s). May be a vector with entries in the range between 0 and 1, inclusive, where alpha = 1 is Lasso and alpha = 0 is Ridge. Defaults to c(0.25, 0.5, 0.75, 1).
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 "auto"). One of "auto", "wAIC", "wBIC", "wGIC", or "wSSE".
auto_ratio_cutoff	Single cutoff for the automatic rule when objective = "auto" (default 1.3). Let r = n_X / p_X, where n_X is the number of training rows and p_X is the number of predictors in the model matrix after dropping the intercept column. If r >= auto_ratio_cutoff, SVEMnet uses wAIC; otherwise it uses wBIC.
gamma	Penalty weight used only when objective = "wGIC" (numeric, default 2). Setting gamma = 2 reproduces wAIC. Larger values approach a BIC-like penalty.
relaxed	Logical, TRUE or FALSE (default TRUE). When TRUE, use glmnet's relaxed elastic net path and select both lambda and relaxed gamma on each bootstrap. When FALSE, fit the standard glmnet path (equivalent to gamma = 1). Note: if relaxed = TRUE and glmnet_alpha includes 0 (ridge), alpha = 0 is dropped.
...	Additional args passed to glmnet() (e.g., penalty.factor, lower.limits, upper.limits, offset, standardize.response, etc.). Any user-supplied weights are ignored so SVEM can supply its own bootstrap weights. Any user-supplied standardize is ignored; SVEMnet always uses standardize = TRUE.

Details

SVEM applies fractional bootstrap weights to training data and anti-correlated weights for validation when weight_scheme = "SVEM". For each bootstrap, glmnet paths are fit for each alpha in

glmnet_alpha, and the lambda (and, if relaxed = TRUE, relaxed gamma) minimizing a weighted validation criterion is selected.

Predictors are always standardized internally via glmnet::glmnet(..., standardize = TRUE).

When relaxed = TRUE, coef(fit, s = lambda, gamma = g) is used to obtain the coefficient path at each relaxed gamma in the internal grid. Metrics are computed from validation-weighted errors and model size is taken as the number of nonzero coefficients including the intercept (support size), keeping selection consistent between standard and relaxed paths. When relaxed = FALSE, gamma is fixed at 1.

Automatic objective rule ("auto"): This function uses a single ratio cutoff, auto_ratio_cutoff, applied to r = n_X / p_X, where p_X is computed from the model matrix with the intercept column removed. If r >= auto_ratio_cutoff the selection criterion is wAIC; otherwise it is wBIC.

Implementation notes for safety:

- The training terms object is stored with environment set to baseenv() to avoid accidental lookups in the calling environment.
- A compact schema (feature names, xlevels, contrasts) is stored to let predict() reconstruct the design matrix deterministically.
- A lightweight sampling schema (numeric ranges and factor levels for raw predictors) is cached to enable random-table generation without needing the original data.

Value

An object of class svem_model with elements:

- parms: averaged coefficients (including intercept).
- parms_debiased: averaged coefficients adjusted by the calibration fit.
- debias_fit: lm(y ~ y_pred) calibration model used for debiasing (or NULL).
- coef_matrix: per-bootstrap coefficient matrix.
- nBoot, glmnet_alpha, best_alphas, best_lambdas, weight_scheme, relaxed.
- best_relax_gammas: per-bootstrap relaxed gamma chosen (NA if relaxed = FALSE).
- objective_input, objective_used, objective (same as objective_used), auto_used, auto_decision, auto_rule, gamma (when wGIC).
- dropped_alpha0_for_relaxed: whether alpha = 0 was removed because relaxed = TRUE.
- schema: list(feature_names, terms_str, xlevels, contrasts, terms_hash) for safe predict.
- sampling_schema: list(predictor_vars, var_classes, num_ranges = rbind(min=..., max=...) for numeric raw predictors, factor_levels = list(...) for factor/character raw predictors).
- diagnostics: list with k_summary (median and IQR of selected size), fallback_rate, n_eff_summary, alpha_freq, relax_gamma_freq.
- actual_y, training_X, y_pred, y_pred_debiased, nobs, nparm, formula, terms, xlevels, contrasts.

Acknowledgments

Development of this package was assisted by GPT o1-preview for structuring parts of the code and documentation.

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-Using/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.

Meinshausen, N. (2007). Relaxed Lasso. Computational Statistics & Data Analysis, 52(1), 374-393.

Examples

```
set.seed(42)

n  <- 30
X1 <- rnorm(n)
X2 <- rnorm(n)
X3 <- rnorm(n)
eps <- rnorm(n, sd = 0.5)
y <- 1 + 2*X1 - 1.5*X2 + 0.5*X3 + 1.2*(X1*X2) + 0.8*(X1^2) + eps
dat <- data.frame(y, X1, X2, X3)

mod_relax <- SVEMnet(
  y ~ (X1 + X2 + X3)^2 + I(X1^2) + I(X2^2),
  data          = dat,
```

```
  glmnet_alpha  = c(1, 0.5),
  nBoot         = 75,
  objective     = "auto",
  weight_scheme = "SVEM",
  relaxed       = FALSE #to run faster to pass CRAN checks
)

pred_in_raw <- predict(mod_relax, dat, debias = FALSE)
pred_in_db  <- predict(mod_relax, dat, debias = TRUE)
```

`svem_random_table_from_model` *Generate a Random Prediction Table from a Fitted SVEMnet Model (no refit)*

Description

Samples the original predictor factor space using ranges and levels cached inside a fitted `svem_model` (from `SVEMnet()`) and computes predictions at those points. Optional mixture groups let you sample composition variables on a (possibly truncated) simplex via a Dirichlet draw. No refitting is performed.

Usage

```
svem_random_table_from_model(
  object,
  n = 1000,
  mixture_groups = NULL,
  debias = FALSE
)
```

Arguments

`object`	A fitted `svem_model` returned by `SVEMnet()`. The object must contain `$sampling_schema`, which is created by the updated `SVEMnet()`.
`n`	Number of random points to generate (default `1000`).
`mixture_groups`	Optional list of mixture factor groups. Each element is a list with components: • `vars`: character vector of mixture variable names (must be in the model). • `lower`: numeric vector of lower bounds (same length as `vars`). • `upper`: numeric vector of upper bounds (same length as `vars`). • `total`: scalar specifying the group total (e.g., 1.0). If omitted, all variables are sampled independently using the cached schema.
`debias`	Logical; if `TRUE`, applies the calibration from `object$debias_fit` (if available) to the mean prediction (default `FALSE`).

Details

This function uses:

- `object$sampling_schema$num_ranges` for uniform sampling of numeric variables.
- `object$sampling_schema$factor_levels` for categorical sampling.
- `object$terms`, `object$xlevels`, and `object$contrasts` (via `predict.svem_model`) to encode the model matrix consistently.

Mixture groups are handled by drawing Dirichlet weights and mapping them to the truncated simplex defined by `lower`, `upper`, and `total`; proposals violating upper bounds are rejected (with oversampling to keep it efficient).

Value

A data frame with sampled predictors and a predicted response column. The response column name matches the left-hand side of `object$formula`.

See Also

SVEMnet, predict.svem_model, svem_significance_test, svem_significance_test_parallel

Examples

```
set.seed(1)
n <- 40
X1 <- runif(n); X2 <- runif(n)
A <- runif(n); B <- runif(n); C <- pmax(0, 1 - A - B)  # simple mixture-ish
F <- factor(sample(c("lo","hi"), n, TRUE))
y <- 1 + 2*X1 - X2 + 3*A + 1.5*B + 0.5*C + (F=="hi") + rnorm(n, 0, 0.3)
d <- data.frame(y, X1, X2, A, B, C, F)

fit <- SVEMnet(y ~ X1 + X2 + A + B + C + F, d, nBoot = 50, glmnet_alpha = 1)

# No mixture constraints (independent sampling using cached ranges/levels)
tbl1 <- svem_random_table_from_model(fit, n = 50)
head(tbl1)

# With mixture constraints for A,B,C that sum to 1 and have bounds
mix <- list(list(vars = c("A","B","C"),
                 lower = c(0.1, 0.1, 0.1),
                 upper = c(0.7, 0.7, 0.7),
                 total = 1.0))
tbl2 <- svem_random_table_from_model(fit, n = 50, mixture_groups = mix)
head(tbl2)
```

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.

Usage

```
svem_significance_test(
  formula,
  data,
  mixture_groups = NULL,
  nPoint = 2000,
  nSVEM = 10,
  nPerm = 150,
  percent = 90,
  nBoot = 100,
  glmnet_alpha = c(1),
  weight_scheme = c("SVEM"),
  objective = c("auto", "wAIC", "wBIC", "wGIC", "wSSE"),
  auto_ratio_cutoff = 1.3,
  gamma = 2,
  relaxed = FALSE,
  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: 10).
nPerm	Number of SVEM fits on permuted responses for the reference distribution (default: 150).
percent	Percentage of variance to capture in the SVD (default: 90).
nBoot	Number of bootstrap iterations within each SVEM fit (default: 100).
glmnet_alpha	The alpha parameter(s) for glmnet (default: c(1)).
weight_scheme	Weighting scheme for SVEM (default: "SVEM").
objective	Objective used inside SVEMnet() to pick the bootstrap path solution. One of "auto", "wAIC", "wBIC", "wGIC", "wSSE" (default: "auto").
auto_ratio_cutoff	Single cutoff for the automatic rule when objective = "auto" (default 1.3). With r = n_X/p_X, if r >= auto_ratio_cutoff use wAIC; else wBIC. Passed to SVEMnet().
gamma	Penalty weight used only when objective = "wGIC" (default 2). Passed to SVEMnet().
relaxed	Logical; default FALSE. When TRUE, inner SVEMnet() fits use glmnet's relaxed elastic net path and select both lambda and relaxed gamma on each bootstrap. When FALSE, the standard glmnet path is used. This value is passed through to SVEMnet(). Note: if relaxed = TRUE and glmnet_alpha includes 0, ridge (alpha = 0) is dropped by SVEMnet() for relaxed fits.
verbose	Logical; if TRUE, displays progress messages (default: TRUE).
...	Additional arguments passed to SVEMnet() and then to glmnet() (for example: penalty.factor, offset, lower.limits, upper.limits, standardize.response, etc.). The relaxed setting is controlled by the relaxed argument of this function and any relaxed value passed via ... is ignored with a warning.

Details

If no mixture groups are supplied, this function behaves identically to a standard SVEM-based whole-model test, sampling non-mixture continuous variables via a maximin Latin hypercube within their observed ranges, and categorical variables from their observed levels.

Internally, predictions at evaluation points use predict.svem_model() with se.fit = TRUE. Rows with unseen categorical levels are returned as NA and are excluded from distance summaries via complete.cases().

Value

A list of class svem_significance_test containing:

- p_value: median p-value across evaluation points.
- p_values: vector of per-point p-values.
- d_Y: distances for original fits.
- d_pi_Y: distances for permutation fits.
- distribution_fit: fitted SHASHo distribution object.
- data_d: data frame combining distances and labels.

See Also

SVEMnet(), predict.svem_model()

---

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 = 10,
  nPerm = 150,
  percent = 90,
  nBoot = 100,
  glmnet_alpha = c(1),
  weight_scheme = c("SVEM"),
  objective = c("auto", "wAIC", "wBIC", "wGIC", "wSSE"),
  auto_ratio_cutoff = 1.3,
  gamma = 2,
  relaxed = FALSE,
  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: 10).
nPerm	Number of SVEM fits on permuted responses for the reference distribution (default: 150).
percent	Percentage of variance to capture in the SVD (default: 90).
nBoot	Number of bootstrap iterations within each SVEM fit (default: 100).
glmnet_alpha	The alpha parameter(s) for glmnet (default: c(1)).
weight_scheme	Weighting scheme for SVEM (default: "SVEM").
objective	Objective used inside SVEMnet() to pick the bootstrap path solution. One of "auto", "wAIC", "wBIC", "wGIC", "wSSE" (default: "auto").
auto_ratio_cutoff	Single cutoff for the automatic rule when objective = "auto" (default 1.3). With r = n_X/p_X, if r >= auto_ratio_cutoff use wAIC; else wBIC. Passed to SVEMnet().
gamma	Penalty weight used only when objective = "wGIC" (default 2). Passed to SVEMnet().
relaxed	Logical; default FALSE. When TRUE, inner SVEMnet() fits use glmnet's relaxed elastic net path and select both lambda and relaxed gamma on each bootstrap. When FALSE, the standard glmnet path is used. This value is passed through to SVEMnet(). Any relaxed provided via ... is ignored with a warning.
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() (for example: penalty.factor, offset, lower.limits, upper.limits, standardize.response, etc.). The relaxed setting is controlled by the relaxed argument of this function and any relaxed value passed via ... is ignored with a warning.

Details

Identical in logic 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

Examples

```
set.seed(1)

# Small toy data with a 3-component mixture A, B, C
n <- 40
sample_trunc_dirichlet <- function(n, lower, upper, total) {
  k <- length(lower)
  stopifnot(length(upper) == k, total >= sum(lower), total <= sum(upper))
  avail <- total - sum(lower)
  if (avail <= 0) return(matrix(rep(lower, each = n), nrow = n))
  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
}

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

mix_spec <- list(list(
  vars  = c("A", "B", "C"),
  lower = lower,
  upper = upper,
  total = total
))

# Parallel significance test (default relaxed = FALSE)
res <- svem_significance_test_parallel(
  y ~ A + B + C + X + F,
  data           = dat,
  mixture_groups = mix_spec,
  glmnet_alpha   = c(1),
  weight_scheme  = "SVEM",
  objective      = "auto",
  auto_ratio_cutoff = 1.3,
  relaxed        = FALSE,   # default, shown for clarity
  nCore          = 2,
  seed           = 123,
  verbose        = FALSE
)
```

```
print(res$p_value)
```

Index

∗ **package**
SVEMnet-package, 2

coef.svem_model, *2*, 3

glmnet_with_cv, *2*, 4

plot.svem_model, *2*, 6
plot.svem_significance_test *(*plot_svem_significance_tests*)*, 7
plot_svem_significance_tests, *2*, 7
predict.svem_cv *(*predict_cv*)*, 9
predict.svem_model, *2*, 7, *15*
predict_cv, 9
print.svem_significance_test, 10

svem_random_table_from_model, 14
svem_significance_test, *2*, *15*, 16
svem_significance_test_parallel, *2*, *15*, 18
SVEMnet, *2*, 10, *15*
SVEMnet-package, 2