Package 'SVEMnet'

September 26, 2025

Type Package

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

Version 2.2.4

Date 2025-09-25

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. (2010) <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>).

Depends R (>= 3.5.0)

Imports glmnet (>= 4.1-2), stats, cluster, ggplot2, lhs, foreach, doParallel, parallel, gamlss, gamlss.dist

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0), withr, vdiffr

VignetteBuilder knitr

License GPL-2 | GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

LazyData true

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2025-09-26 07:20:10 UTC

Contents

`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 (optionally debiased, with intervals when available).

`coef.svem_model` Bootstrap nonzero percentages and summary of coefficients.

`plot.svem_model` Quick actual-versus-predicted plot for a fitted model.

`bigexp_terms` Build a deterministic expanded RHS (polynomials, interactions) with locked levels/ranges.

`bigexp_formula` Reuse a locked expansion for another response to ensure identical factor space.

svem_random_table_multi Generate one shared random predictor table (with optional mixture constraints) and get predictions from multiple SVEM models at those points.

svem_optimize_random Random-search optimizer for multiple responses with goals, weights, optional CIs, and diverse PAM-medoids candidates.

svem_significance_test Whole-model significance test with optional mixture-group sampling.

svem_significance_test_parallel Parallel whole-model significance test (foreach + doParallel).

plot_svem_significance_tests Plot helper for visualizing multiple significance-test outputs.

glmnet_with_cv Convenience wrapper around repeated `cv.glmnet()` selection.

lipid_screen Example dataset for multi-response modeling and mixture-constrained optimization demos.

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*.

Friedman, J. H., Hastie, T., and 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.

`bigexp_formula` *Construct a formula for a new response using a bigexp_spec*

Description

Useful when you want to fit multiple responses with the same expansion.

Usage

```
bigexp_formula(spec, response)
```

Arguments

`spec`	A "bigexp_spec".
`response`	Character scalar: the new response (column name in your data).

Value

A formula like `response ~ <locked big expansion>`.

Examples

```
# f2 <- bigexp_formula(spec, "y2")
```

bigexp_model_matrix *Build a model matrix using the spec's stored contrasts (if present)*

Description

Build a model matrix using the spec's stored contrasts (if present)

Usage

```
bigexp_model_matrix(spec, data)
```

Arguments

spec	A "bigexp_spec".
data	A data frame to prepare and encode.

Value

The design matrix returned by `model.matrix()`.

Examples

```
# MM <- bigexp_model_matrix(spec, newdata)
```

bigexp_prepare *Prepare data to match a bigexp_spec (stable expansion across datasets)*

Description

Coerces new data to the types/levels locked in `spec` so that `model.matrix(spec$formula, data)` yields the same columns and order every time, even if some factor levels are absent in the new batch.

Usage

```
bigexp_prepare(spec, data, unseen = c("warn_na", "error"))
```

Arguments

spec	Object returned by `bigexp_terms()`.
data	New data frame (train/test/future batches).
unseen	How to handle unseen factor levels in `data`: `"warn_na"` (default; convert to `NA` with a warning) or `"error"` (stop).

Value

`list(formula = spec$formula, data = coerced_data)`

Examples

```
# spec <- bigexp_terms(y ~ X1 + X2 + G, train)
# new_in <- bigexp_prepare(spec, newdata)
# fit <- some_model(new_in$formula, data = new_in$data, ...)
```

`bigexp_terms` *Create a deterministic expansion spec for wide polynomial/interaction models*

Description

Builds a specification that locks variable types and factor levels from `data`, then encodes a large expansion:

- Full factorial up to 2- or 3-way among the listed main effects
- Response surface (squares of continuous predictors)
- Partial cubic crosses (I(X^2):Z, optionally 3-way I(X^2):Z:W)

Usage

```
bigexp_terms(
  formula,
  data,
  factorial_order = 3L,
  discrete_threshold = 2L,
  include_pure_cubic = FALSE,
  include_pc_3way = FALSE,
  intercept = TRUE
)
```

Arguments

`formula`	Main-effects formula on the RHS (no `:/^/I()` there). `y ~ .` allowed.
`data`	Data frame used to decide types and lock factor levels.
`factorial_order`	Integer, 2 or 3 (default 3).
`discrete_threshold`	Numeric predictors with <= this many unique finite values are treated as categorical (default 2).
`include_pure_cubic`	Logical; include `I(X^3)` for continuous predictors (default FALSE).
`include_pc_3way`	Logical; include partial-cubic 3-ways `I(X^2):Z:W` (default FALSE).
`intercept`	Include intercept (default TRUE).

Details

Provide a main-effects formula (e.g., `y ~ X1 + X2 + G` or `y ~ .`). The function constructs the complex RHS and records factor levels so that future datasets expand identically.

Value

An object of class `"bigexp_spec"` with components:

- `formula` — expanded formula (`y ~ ...`) using the training response.
- `rhs` — the right-hand-side expansion string, reusable for any response.
- `vars` — predictor names (in the order discovered from `formula`/`data`).
- `is_cat` — named logical: categorical vs continuous.
- `levels` — list of locked factor levels (level order preserved).
- `num_range` — 2 x p matrix of ranges for continuous vars (informational).
- `settings` — list of expansion settings, including saved contrasts.

Examples

```
# spec <- bigexp_terms(y ~ X1 + X2 + G, data = df, factorial_order = 3)
```

`bigexp_train` *Build a spec and prepare training data in one call*

Description

Convenience wrapper around `bigexp_terms()` and `bigexp_prepare()`. It creates the deterministic expansion spec from the training data and immediately prepares that same training data to match the locked types/levels. Prefer the two-step API in documentation, but this is handy for quick scripts.

Usage

```
bigexp_train(formula, data, ...)
```

Arguments

`formula`	Main-effects formula (e.g., `y ~ X1 + X2 + G` or `y ~ .`).
`data`	Training data frame (used to lock types/levels and prepare).
`...`	Additional arguments forwarded to `bigexp_terms()` (e.g., `factorial_order`, `discrete_threshold`, `include_pure_cubic`, `include_pc_3way`, `intercept`).

Value

A list with components:

- spec — the "bigexp_spec" object.
- formula — the expanded formula (spec$formula).
- data — the coerced training data ready for modeling.

Examples

```
# tr <- bigexp_train(y ~ X1 + X2 + X3, dat, factorial_order = 3)
# fit <- SVEMnet(tr$formula, data = tr$data, ...)
```

---

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, plot = TRUE, print_table = TRUE, ...)
```

Arguments

object	An object of class svem_model.
tol	Numeric tolerance for "nonzero" (default 1e-7).
plot	Logical; if TRUE, draws a quick ggplot summary (default TRUE).
print_table	Logical; if TRUE, prints a compact table (default TRUE).
...	Unused.

Value

Invisibly returns a data frame with columns: Variable, Percent of Bootstraps Nonzero.

`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)
```

`lipid_screen` *Lipid formulation screening data*

Description

An example dataset for modeling Potency, Size, and PDI as functions of formulation and process settings. Percent composition columns are stored as proportions in 0, 1 (e.g., 4.19% is 0.0419). This table is intended for demonstration of SVEMnet multi-response modeling and random-search optimization.

Usage

```
lipid_screen
```

Format

A data frame with N rows and the following columns:

RunID character. Optional identifier.

PEG numeric. Proportion (0–1).

Helper numeric. Proportion (0–1).

Ionizable numeric. Proportion (0–1).

Cholesterol numeric. Proportion (0–1).

Ionizable_Lipid_Type factor.

N_P_ratio numeric.

flow_rate numeric.

Potency numeric. Response.

Size numeric. Response (e.g., nm).

PDI numeric. Response (polydispersity index).

Notes character. Optional.

Details

This dataset accompanies examples showing:

- fitting three SVEM models (Potency, Size, PDI) on a shared expanded basis,
- generating shared random predictor points with `svem_random_table_multi()`,
- optimizing a weighted multi-response goal with `svem_optimize_random()`.

Source

Simulated screening table supplied by the package author.

Examples

```
library(SVEMnet)

# 1) Load the bundled dataset
data(lipid_screen)
str(lipid_screen)

# 2) Build a deterministic expansion using bigexp_terms()
#    Provide main effects only on the RHS; expansion width is controlled via arguments.
spec <- bigexp_terms(
  Potency ~ PEG + Helper + Ionizable + Cholesterol +
    Ionizable_Lipid_Type + N_P_ratio + flow_rate,
  data              = lipid_screen,
  factorial_order   = 3,       # up to 3-way interactions
  include_pure_cubic = TRUE,
  include_pc_3way   = FALSE
)

# 3) Reuse the same locked expansion for other responses
form_pot <- bigexp_formula(spec, "Potency")
form_siz <- bigexp_formula(spec, "Size")
form_pdi <- bigexp_formula(spec, "PDI")

# 4) Fit SVEM models with the shared factor space/expansion
set.seed(1)
fit_pot <- SVEMnet(form_pot, lipid_screen, nBoot = 60, glmnet_alpha = 1, relaxed = TRUE)
fit_siz <- SVEMnet(form_siz, lipid_screen, nBoot = 60, glmnet_alpha = 1, relaxed = TRUE)
fit_pdi <- SVEMnet(form_pdi, lipid_screen, nBoot = 60, glmnet_alpha = 1, relaxed = TRUE)

# 5) Collect models in a named list by response
objs <- list(Potency = fit_pot, Size = fit_siz, PDI = fit_pdi)

# 6) Define multi-response goals and weights
#    Maximize Potency (0.7), minimize Size (0.2), minimize PDI (0.1)
goals <- list(
  Potency = list(goal = "max", weight = 0.7),
  Size    = list(goal = "min", weight = 0.2),
  PDI     = list(goal = "min", weight = 0.1)
)
```

```
# Mixture constraints: components sum to 1, with bounds
mix <- list(list(
  vars  = c("PEG", "Helper", "Ionizable", "Cholesterol"),
  lower = c(0.01, 0.10, 0.10, 0.10),
  upper = c(0.05, 0.60, 0.60, 0.60),
  total = 1.0
))

opt_out <- svem_optimize_random(
  objects        = objs,
  goals          = goals,
  n              = 5000,
  mixture_groups = mix,
  agg            = "mean",
  debias         = FALSE,
  ci             = TRUE,
  level          = 0.95,
  k_candidates   = 5,
  top_frac       = 0.01,
  verbose        = TRUE
)

# Inspect results
opt_out$best_x
opt_out$best_pred
opt_out$best_ci
opt_out$candidates
```

`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_test` *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, with optional bootstrap standard errors and percentile confidence intervals.

Usage

```
## S3 method for class 'svem_model'
predict(
  object,
  newdata,
  debias = FALSE,
  se.fit = FALSE,
  interval = FALSE,
  level = 0.95,
  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 based on the bootstrap spread when member predictions are available (default FALSE).
interval	Logical; if TRUE, returns percentile confidence limits based on bootstrap predictions when available (default FALSE).
level	Confidence level for the percentile interval. Default 0.95.
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 follows these rules:

- The set of columns produced by model.matrix on newdata is aligned to those used in training.
- Any training columns dropped by model.matrix are added back as zeros.
- Columns are reordered to the exact training order before multiplication.

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

When agg = "mean", se.fit is the row-wise sd of member predictions. If debias = TRUE and a finite calibration slope is available, both member predictions and their sd are transformed by the calibration before aggregation. Percentile intervals are computed from the transformed member predictions.

Value

A numeric vector of predictions, or a list with components as follows:

- fit: predictions
- se.fit: bootstrap standard errors when se.fit = TRUE
- lwr, upr: percentile confidence limits when interval = 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)

# Mean aggregation with SEs and 95 percent CIs
out <- predict(fit, dat, debias = TRUE, se.fit = TRUE, interval = 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)
```

`predict_with_ci` *Percentile Confidence Intervals for SVEM Predictions*

Description

Computes predictions and percentile confidence intervals from bootstrap member predictions for a fitted svem_model.

Usage

```
predict_with_ci(object, newdata, level = 0.95, debias = FALSE)
```

Arguments

`object`	A svem_model.
`newdata`	Data frame of predictors.
`level`	Confidence level (default 0.95).
`debias`	Logical; if TRUE, apply calibration to each member prediction (default FALSE).

Value

A data.frame with columns fit, lwr, upr.

`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", "wSSE"),
  auto_ratio_cutoff = 1.3,
  relaxed = TRUE,
  response = NULL,
  unseen = c("warn_na", "error"),
  ...
)
```

Arguments

formula	A formula specifying the model to be fitted, OR a bigexp_spec created by bigexp_terms().
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", 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.
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. Note: if relaxed = TRUE and glmnet_alpha includes 0 (ridge), alpha = 0 is dropped.
response	Optional character. When formula is a bigexp_spec, this names the response column to use on the LHS; defaults to the response stored in the spec.
unseen	How to treat unseen factor levels when formula is a bigexp_spec: "warn_na" (default; convert to NA with a warning) or "error" (stop).
...	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

You can pass either:

- a standard model formula, e.g. y ~ X1 + X2 + X3 + I(X1^2) + (X1 + X2 + X3)^2
- a bigexp_spec created by bigexp_terms(), in which case SVEMnet will prepare the data deterministically (locked types/levels) and, if requested, swap the response to fit multiple independent responses over the same expansion.

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.

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.
- 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.
- used_bigexp_spec: logical flag indicating whether a bigexp_spec was used.

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-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., and 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)
```

```
# Minimal hand-written expansion
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
)

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


# ------------------------------------------------------------------------------
# Big expansion (full factorial + response surface + partial cubic)
# Build once, reuse for one or more responses
# ------------------------------------------------------------------------------
spec <- bigexp_terms(
  y ~ X1 + X2 + X3, data = dat,
  factorial_order    = 3,      # allow 3-way factorials
  include_pc_3way    = FALSE,  # set TRUE to add I(X^2):Z:W
  include_pure_cubic = FALSE
)

# Fit using the spec (auto-prepares data)
fit_y <- SVEMnet(
  spec, dat,
  glmnet_alpha  = c(1, 0.5),
  nBoot         = 50,
  objective     = "auto",
  weight_scheme = "SVEM",
  relaxed       = FALSE
)

# A second, independent response over the same expansion
set.seed(99)
dat$y2 <- 0.5 + 1.4*X1 - 0.6*X2 + 0.2*X3 + rnorm(n, 0, 0.4)
fit_y2 <- SVEMnet(
  spec, dat, response = "y2",
  glmnet_alpha  = c(1, 0.5),
  nBoot         = 50,
  objective     = "auto",
  weight_scheme = "SVEM",
  relaxed       = FALSE
)

p1  <- predict(fit_y,  dat)
p2  <- predict(fit_y2, dat, debias = TRUE)

# Show that a new batch expands identically under the same spec
newdat <- data.frame(
```

```
  y  = y,
  X1 = X1 + rnorm(n, 0, 0.05),
  X2 = X2 + rnorm(n, 0, 0.05),
  X3 = X3 + rnorm(n, 0, 0.05)
)
prep_new <- bigexp_prepare(spec, newdat)
stopifnot(identical(
  colnames(model.matrix(spec$formula, bigexp_prepare(spec, dat)$data)),
  colnames(model.matrix(spec$formula, prep_new$data))
))
preds_new <- predict(fit_y, prep_new$data)
```

`svem_optimize_random` *Random-Search Optimizer with Goals, Weights, Optional CIs, and Diverse Candidates*

Description

Draws random points via `svem_random_table_multi`, scores them using user goals and weights, returns the best design point, and optionally proposes `k_candidates` diverse high-score candidates by clustering the top fraction of rows with Gower distance and PAM medoids. Medoids are representative existing rows, so proposed candidates are guaranteed to be feasible under all sampling and mixture constraints.

Usage

```
svem_optimize_random(
  objects,
  goals,
  n = 10000,
  mixture_groups = NULL,
  debias = FALSE,
  agg = c("parms", "mean"),
  ci = TRUE,
  level = 0.95,
  top_frac = 0.01,
  k_candidates = 0,
  verbose = TRUE
)
```

Arguments

`objects`	Named list of `svem_model` objects (from `SVEMnet()`). List names must match the response names (left-hand sides) of the models.
`goals`	Named list per response of the form `list(goal = "max"|"min"|"target", weight = nonnegative number, target = number when goal = "target")`. Weights are normalized to sum to one internally.

n	Number of random samples to draw.
mixture_groups	Optional mixture constraints forwarded to svem_random_table_multi(). Each group may include vars, lower, upper, and total.
debias	Logical; if TRUE, use debiased predictions for scoring where available.
agg	Aggregation for point predictions, one of "parms" or "mean". This is passed to predict.svem_model when applicable.
ci	Logical; if TRUE, compute percentile confidence intervals when available via predict(..., interval = TRUE) or predict_with_ci(...).
level	Confidence level for percentile intervals (default 0.95).
top_frac	Fraction (0, 1] of highest-score rows to cluster (default 0.01).
k_candidates	Integer number of diverse candidates (medoids) to return (default 0). If 0, no clustering is performed.
verbose	Logical; print a compact summary of the run and results.

Value

A list with:

- best_idx: Row index of the selected best design in the sampled table.
- best_x: Predictors at the best design.
- best_pred: Named numeric vector of predicted responses at best_x.
- best_ci: Data frame of percentile limits when ci = TRUE; otherwise NULL.
- candidates: Data frame of k_candidates diverse candidates (medoids; existing rows) with predictors, predictions, optional CIs, and score; NULL if k_candidates = 0.
- score_table: Sampled table with response columns, normalized and weighted contributions, and final score.
- weights: Normalized weights used in scoring.
- goals: Tidy data frame describing each response goal, weight, and target.

Scoring

Each response is normalized on the sampled range and combined via a weighted sum:

- "max": normalize01(y)
- "min": normalize01(-y)
- "target": normalize01(-abs(y - target))

The normalize01 function maps to [0,1] using the sampled minimum and maximum.

Diverse candidates

We take the top top_frac fraction by score, compute Gower distances on predictors, and run PAM to get medoids. Returning medoids rather than centroids ensures each candidate corresponds to an actual sampled setting that satisfies constraints.

Examples

```
set.seed(1)
n  <- 120
X1 <- runif(n); X2 <- runif(n)
F  <- factor(sample(c("lo","hi"), n, TRUE))
y1 <- 1 + 1.5*X1 - 0.8*X2 + 0.4*(F=="hi") + rnorm(n, 0, 0.2)
y2 <- 0.7 + 0.4*X1 + 0.4*X2 - 0.2*(F=="hi") + rnorm(n, 0, 0.2)
dat <- data.frame(y1, y2, X1, X2, F)

m1 <- SVEMnet(y1 ~ X1 + X2 + F, dat, nBoot = 30)
m2 <- SVEMnet(y2 ~ X1 + X2 + F, dat, nBoot = 30)
objs <- list(y1 = m1, y2 = m2)

goals <- list(
  y1 = list(goal = "max",    weight = 0.6),
  y2 = list(goal = "target", weight = 0.4, target = 0.9)
)

out <- svem_optimize_random(
  objects      = objs,
  goals        = goals,
  n            = 3000,
  agg          = "mean",
  debias       = FALSE,
  ci           = TRUE,
  level        = 0.95,
  k_candidates = 5,
  top_frac     = 0.02,
  verbose      = TRUE
)
out$best_x; head(out$candidates)

# Mixture-constrained lipid example (composition sums to 1)
data(lipid_screen)
spec <- bigexp_terms(
  Potency ~ PEG + Helper + Ionizable + Cholesterol +
    Ionizable_Lipid_Type + N_P_ratio + flow_rate,
  data = lipid_screen, factorial_order = 3
)
fP <- bigexp_formula(spec, "Potency")
fS <- bigexp_formula(spec, "Size")
fD <- bigexp_formula(spec, "PDI")
mP <- SVEMnet(fP, lipid_screen, nBoot = 40)
mS <- SVEMnet(fS, lipid_screen, nBoot = 40)
mD <- SVEMnet(fD, lipid_screen, nBoot = 40)
objs2 <- list(Potency = mP, Size = mS, PDI = mD)

goals2 <- list(
  Potency = list(goal = "max", weight = 0.7),
  Size    = list(goal = "min", weight = 0.2),
  PDI     = list(goal = "min", weight = 0.1)
)
```

```
mixL <- list(list(
  vars  = c("Cholesterol","PEG","Ionizable","Helper"),
  lower = c(0.10, 0.01, 0.10, 0.10),
  upper = c(0.60, 0.05, 0.60, 0.60),
  total = 1
))

opt <- svem_optimize_random(
  objects        = objs2,
  goals          = goals2,
  n              = 8000,
  mixture_groups = mixL,
  agg            = "mean",
  debias         = FALSE,
  ci             = TRUE,
  level          = 0.95,
  k_candidates   = 5,
  top_frac       = 0.01,
  verbose        = TRUE
)
opt$best_x; head(opt$candidates)
```

`svem_random_table_multi` *Generate a Random Prediction Table from Multiple SVEMnet Models (no refit)*

Description

Samples the original predictor factor space cached in fitted `svem_model` objects and computes predictions from each model at the same random points. Intended for multiple responses built over the same factor space and a deterministic factor expansion.

Usage

```
svem_random_table_multi(
  objects,
  n = 1000,
  mixture_groups = NULL,
  debias = FALSE,
  range_tol = 1e-08
)
```

Arguments

objects	A list of fitted svem_model objects returned by SVEMnet(). Each object must contain $sampling_schema produced by the updated SVEMnet(). A single model is also accepted and treated as a length-one list.
n	Number of random points to generate. Default is 1000.
mixture_groups	Optional list of mixture constraint groups. Each group is a list with elements vars, lower, upper, total. The variables in vars must be numeric-like predictors present in all models. The sampler uses a truncated Dirichlet so that elementwise bounds are respected and sum(vars) = total.
debias	Logical; if TRUE, apply each model's calibration during prediction when available. Default is FALSE.
range_tol	Numeric tolerance for comparing numeric ranges across models. Default is 1e-8.

Details

All models must share an identical predictor schema:

- The same predictor_vars in the same order
- The same var_classes for each predictor
- Identical factor levels and level order
- Numeric ranges that match within range_tol

The function stops with an informative error message if any of these checks fail.

Value

A list with three data frames:

- data: the sampled predictor settings, one row per random point.
- pred: one column per response, aligned to data rows.
- all: cbind(data, pred) for convenience.

Each prediction column is named by the model's response (left-hand side). If a response name would collide with a predictor name, ".pred" is appended.

Sampling strategy

Non-mixture numeric variables are sampled with a maximin Latin hypercube over the cached numeric ranges. Mixture variables are sampled jointly within each specified group using a truncated Dirichlet so that elementwise bounds and the total sum are satisfied. Categorical variables are sampled from the cached factor levels. The same random predictor table is fed to each model so response columns are directly comparable.

Notes on mixtures

Each mixture group should list only numeric-like variables. Bounds are interpreted on the original scale of those variables. If total equals the sum of lower bounds, the sampler returns the lower-bound corner for that group.

See Also

SVEMnet, predict.svem_model

Examples

```
set.seed(1)
n <- 60
X1 <- runif(n); X2 <- runif(n)
A <- runif(n); B <- runif(n); C <- pmax(0, 1 - A - B)
F <- factor(sample(c("lo","hi"), n, TRUE))
y1 <- 1 + 2*X1 - X2 + 3*A + 1.5*B + 0.5*C + (F=="hi") + rnorm(n, 0, 0.3)
y2 <- 0.5 + 0.8*X1 + 0.4*X2 + rnorm(n, 0, 0.2)
d  <- data.frame(y1, y2, X1, X2, A, B, C, F)

fit1 <- SVEMnet(y1 ~ X1 + X2 + A + B + C + F, d, nBoot = 40)
fit2 <- SVEMnet(y2 ~ X1 + X2 + A + B + C + F, d, nBoot = 40)

# Mixture constraint for A, B, C that sum to 1
mix <- list(list(vars=c("A","B","C"),
                 lower=c(0,0,0), upper=c(1,1,1), total=1))

res <- svem_random_table_multi(
  list(fit1, fit2), n = 200, mixture_groups = mix,
  debias = FALSE
)
head(res$all)

# Lipid screening example with composition group
data(lipid_screen)
spec <- bigexp_terms(
  Potency ~ PEG + Helper + Ionizable + Cholesterol +
    Ionizable_Lipid_Type + N_P_ratio + flow_rate,
  data = lipid_screen, factorial_order = 3
)
fP <- bigexp_formula(spec, "Potency")
fS <- bigexp_formula(spec, "Size")
fD <- bigexp_formula(spec, "PDI")
mP <- SVEMnet(fP, lipid_screen, nBoot = 30)
mS <- SVEMnet(fS, lipid_screen, nBoot = 30)
mD <- SVEMnet(fD, lipid_screen, nBoot = 30)

mixL <- list(list(
  vars  = c("Cholesterol","PEG","Ionizable","Helper"),
  lower = c(0.10, 0.01, 0.10, 0.10),
  upper = c(0.60, 0.05, 0.60, 0.60),
  total = 1
))

tab <- svem_random_table_multi(
  objects        = list(Potency = mP, Size = mS, PDI = mD),
  n              = 1000,
  mixture_groups = mixL,
```

```
  debias         = FALSE
)
head(tab$all)
```

`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", "wSSE"),
  auto_ratio_cutoff = 1.3,
  relaxed = FALSE,
  verbose = FALSE,
  spec = NULL,
  response = NULL,
  use_spec_contrasts = TRUE,
  ...
)
```

Arguments

`formula`	A formula specifying the model to be tested. If `spec` is provided, the right-hand side is ignored and replaced by the locked expansion in `spec`.

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", "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().
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: FALSE).
spec	Optional bigexp_spec created by bigexp_terms(). If provided, the test reuses its locked expansion. The working formula becomes bigexp_formula(spec, response_name), where response_name is taken from response if supplied, otherwise from the left-hand side of formula. Categorical sampling uses spec$levels and numeric sampling prefers spec$num_range when available.
response	Optional character name for the response variable to use when spec is supplied. If omitted, the response is taken from the left-hand side of formula.
use_spec_contrasts	Logical; default TRUE. When spec is supplied and use_spec_contrasts=TRUE, the function temporarily replays spec$settings$contrasts_options for deterministic coding, then restores the caller's options on exit.
...	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

This function can optionally reuse a deterministic, locked expansion built with `bigexp_terms()`. Provide `spec` (and optionally `response`) to ensure that categorical levels, contrasts, and the polynomial/interaction structure are identical across repeated calls and across multiple responses of the same factor space.

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()`.

When reusing a `spec`, you can fit many responses independently over the same encoded factor space by calling this function repeatedly with different `response` values.

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.

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., and 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.

See Also

`SVEMnet()`, `predict.svem_model()`, `bigexp_terms()`, `bigexp_formula()`

`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", "wSSE"),
  auto_ratio_cutoff = 1.3,
  relaxed = FALSE,
  verbose = TRUE,
  nCore = parallel::detectCores(),
```

```
  seed = NULL,
  spec = NULL,
  response = NULL,
  use_spec_contrasts = TRUE,
  ...
)
```

Arguments

formula	A formula specifying the model to be tested. If spec is provided, the right-hand side is ignored and replaced by the locked expansion in spec.
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.
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", "wSSE" (default: "auto"). (Note: "wGIC" is no longer supported.)
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().
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).
nCore	Number of CPU cores for parallel processing (default: all available cores).
seed	Optional integer seed for reproducible parallel RNG (default: NULL).

spec — Optional bigexp_spec created by bigexp_terms(). If provided, the test reuses its locked expansion. The working formula becomes bigexp_formula(spec, response_name), where response_name is taken from response if supplied, otherwise from the left-hand side of formula. Categorical sampling uses spec$levels and numeric sampling prefers spec$num_range when available.

response — Optional character name for the response variable to use when spec is supplied. If omitted, the response is taken from the left-hand side of formula.

use_spec_contrasts — Logical; default TRUE. When spec is supplied and use_spec_contrasts=TRUE, the function replays spec$settings$contrasts_options on the parallel workers for deterministic coding.

... — 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.

This parallel version can optionally reuse a deterministic, locked expansion built with bigexp_terms(). Provide spec (and optionally response) to ensure that categorical levels, contrasts, and the polynomial/interaction structure are identical across repeated calls and across multiple responses of the same factor space.

Value

A list of class svem_significance_test containing the test results.

See Also

svem_significance_test, bigexp_terms, bigexp_formula

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)
```

`with_bigexp_contrasts` *Evaluate an expression with the spec's recorded contrast options*

Description

Evaluate an expression with the spec's recorded contrast options

Usage

```
with_bigexp_contrasts(spec, code)
```

Arguments

spec	A "bigexp_spec".
code	Code to evaluate with temporarily restored options(contrasts=...).

Examples

```
# with_bigexp_contrasts(spec, { fit <- SVEMnet(f, data = d, ...) })
```

Index