Package 'psAve'

July 10, 2026

Type Package

Title Model-Averaged Propensity Scores Selected by Prognostic-Score Balance

Version 1.0.1

Description Constructs a model-averaged propensity score as a convex combination of candidate propensity score models, with mixing weights selected on a simplex grid to optimize covariate or prognostic-score balance, implementing the method of Kabata, Stuart and Shintani (2024) <doi:10.1186/s12874-024-02350-y>. Prognostic scores follow Hansen (2008) <doi:10.1093/biomet/asn004>: outcome models are fit on untreated units only. The resulting score is designed to be supplied directly to the matchit() function of 'MatchIt' as a distance measure or to the weightit() function of 'WeightIt' as a propensity score, with balance assessment via 'cobalt'.

Depends R (>= 4.1)

Imports cobalt (>= 4.6.0), stats, utils, graphics

Suggests MatchIt, WeightIt, SuperLearner, rpart, ranger, xgboost, survey, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

License GPL (>= 2)

URL https://kabajiro.github.io/psAve/, https://github.com/kabajiro/psAve

BugReports https://github.com/kabajiro/psAve/issues

Encoding UTF-8

RoxygenNote 8.0.0

NeedsCompilation no

Author Daijiro Kabata [aut, cre, cph]

Maintainer Daijiro Kabata <daijiro.kabata@port.kobe-u.ac.jp>

Repository CRAN

Date/Publication 2026-07-10 19:10:02 UTC

Contents

`bal.tab.psave` *Balance tables for psave objects*

Description

A method for `cobalt::bal.tab()`: assesses balance on the covariates of a `psave()` fit under the implied inverse-probability weights, with the model-averaged propensity score and (when available) the model-averaged prognostic score supplied as `distance` measures – the prognostic-score balance diagnostic of Stuart, Lee and Leacy (2013).

Usage

```
## S3 method for class 'psave'
bal.tab(x, ...)
```

Arguments

`x`	A `psave` object.
`...`	Further arguments passed on to `cobalt::bal.tab()` (e.g., `un = TRUE`, `thresholds = c(m = 0.1)`).

Details

The call delegates to the default **cobalt** machinery as `cobalt::bal.tab(<covariates>, treat = x$treat, weights = x$w`
so all the usual **cobalt** arguments (`un`, `stats`, `thresholds`, ...) are available, and display conventions are **cobalt**'s own (the *selection criterion* inside `psave()` uses the paper's uniform sample-SD standardization instead; see `psave_criteria()`).

Value

A `bal.tab` object; see `cobalt::bal.tab()`.

References

Stuart EA, Lee BK, Leacy FP (2013). Prognostic score-based balance measures can be a useful diagnostic for propensity score methods in comparative effectiveness research. *Journal of Clinical Epidemiology*, 66(8), S84-S90. doi:10.1016/j.jclinepi.2013.01.013

See Also

`psave()`, `cobalt::bal.tab()`, `plot.psave()`

Examples

```
data("lalonde", package = "MatchIt")
fit <- psave(treat ~ age + educ + married + re74, data = lalonde,
             outcome = ~ re78, ps.methods = "glm", prog.methods = "glm")
cobalt::bal.tab(fit, un = TRUE)
```

`fitted.psave` *Extract the averaged propensity or prognostic score*

Description

`fitted()` is the canonical extractor for the model-averaged scores of a `psave()` fit; `weights()` extracts the inverse-probability weights implied by the averaged propensity score at the fitted estimand.

Usage

```
## S3 method for class 'psave'
fitted(object, type = c("ps", "prog"), ...)

## S3 method for class 'psave'
weights(object, ...)
```

Arguments

`object`	A psave object.
`type`	`"ps"` (default) for the model-averaged propensity score, or `"prog"` for the model-averaged prognostic score.
`...`	Ignored.

Value

For `fitted()`, a numeric vector named by the rownames of the analyzed data. For `weights()`, the numeric vector `object$weights` (weights at the *fitted* estimand only; for other estimands use `WeightIt::get_w_from_ps(fitted(object), object$treat, estimand = ...)`).

See Also

psave(), predict.psave()

---

plot.psave *Plot a psave object*

Description

Three diagnostic displays for a psave() fit:

"balance" a Love plot of covariate and prognostic-score balance before/after weighting, via cobalt::love.plot() (dispatched through bal.tab.psave(); **cobalt** is an Import, so always available).

"distribution" the distribution of the propensity scores by treatment group: grey curves for the candidate models, a colored curve for the selected average (base graphics). Inspect this for extreme candidate scores.

"criterion" the selection criterion over the λ grid: exact for up to three candidates (a curve for M = 2, a colored grid map for M = 3); for M > 3, one profile per candidate (the minimum criterion value attainable at each value of λ_m). Requires the stored path (keep.path = TRUE and a grid of at most 100,000 rows).

Usage

```
## S3 method for class 'psave'
plot(x, type = c("balance", "distribution", "criterion"), ...)
```

Arguments

x	A psave object.
type	One of "balance" (default), "distribution", "criterion".
...	For "balance", further arguments to cobalt::love.plot() (e.g., thresholds = 0.1); otherwise further graphical parameters passed to the base plotting calls.

Value

For "balance", the ggplot object from cobalt::love.plot() (invisibly, after printing); otherwise x, invisibly.

See Also

psave(), bal.tab.psave(), cobalt::love.plot()

predict.psave *Predict averaged scores for new data*

Description

Computes the model-averaged propensity score (or prognostic score) for new observations by applying the stored candidate fits to `newdata`, clipping candidate propensity scores as at fit time, and combining them with the selected mixing weights. Requires `psave(..., keep.fits = TRUE)`.

Usage

```
## S3 method for class 'psave'
predict(object, newdata, type = c("ps", "prog"), ...)
```

Arguments

`object`	A `psave` object fitted with `keep.fits = TRUE`.
`newdata`	A data frame containing the variables of the propensity score formula (for `type = "ps"`) or of the prognostic specification (for `type = "prog"`). Missing values are an error.
`type`	`"ps"` (default) or `"prog"`.
`...`	Ignored.

Value

A numeric vector with one score per row of `newdata`, named by its rownames. If `newdata` is missing, the in-sample fitted scores are returned (equivalent to `fitted.psave()`).

See Also

`psave()`, `fitted.psave()`

print.psave *Print a psave object*

Description

Prints a one-screen summary of a fitted `psave()` object: estimand and criterion, the selected mixing weights λ and γ as labeled text bars, the criterion value, a three-row balance preview (the worst covariates plus the prognostic score), and then the **literal next call** – echoing the formula and data name from your own `psave()` call – that hands the averaged score to `MatchIt::matchit()` or `WeightIt::weightit()`.

Usage

```
## S3 method for class 'psave'
print(x, digits = 3, ...)
```

Arguments

x	A psave object.
digits	Number of significant digits to print. Default 3.
...	Ignored.

Value

x, invisibly.

See Also

psave(), summary.psave()

---

psave *Model-averaged propensity scores selected by prognostic-score balance*

Description

psave() constructs a model-averaged propensity score $\bar{e}(X) = \sum_m \lambda_m \hat{e}_m(X)$: a convex combination of candidate propensity score models whose mixing weights λ are selected on a simplex grid to optimize a balance criterion – by default the weighted absolute standardized mean difference of a **model-averaged prognostic score** (the "Prog (Ave)" estimator of Kabata, Stuart and Shintani 2024). The result is deliberately modest: a numeric score vector designed to be handed to MatchIt::matchit() as distance, or to WeightIt::weightit() as ps, with balance assessment via **cobalt**.

Usage

```
psave(
  formula,
  data,
  outcome = NULL,
  estimand = c("ATT", "ATE"),
  criterion = c("prog", "smd", "ks", "logloss"),
  prog.target = "average",
  ps.methods = c("glm", "rpart", "ranger", "xgboost"),
  prog.methods = c("glm", "rpart", "ranger", "xgboost"),
  ps.matrix = NULL,
  prog.matrix = NULL,
  ps.append = NULL,
  prog.append = NULL,
```

```
  average = TRUE,
  family = gaussian(),
  step = 0.05,
  clip = c(0.01, 0.99),
  s.d.denom = "treated",
  cv = 5L,
  control = list(),
  keep.fits = FALSE,
  keep.path = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`formula`	A two-sided formula `treat ~ x1 + x2 + ...`, exactly as in `MatchIt::matchit()`. The right-hand side defines both the candidate-PS covariates and the balance covariates used by the `smd`/`ks` criteria.
`data`	A data frame containing the variables in `formula` (and `outcome`). Complete cases in all used variables are REQUIRED; any missing value is an error, never a silent row drop.
`outcome`	The outcome specification for the prognostic score: a one-sided formula `~ y` (the formula right-hand side is reused as the prognostic predictors) or a two-sided formula `y ~ z1 + z2` (a distinct prognostic specification). Required when `criterion = "prog"`; optional for the outcome-free criteria (`"smd"`, `"ks"`, `"logloss"`), where it may still be supplied so that the prognostic score is estimated and reported in `diagnostics`, `balance`, and `bal.tab.psave()`. Prognostic models are fit on **untreated units only** (Hansen 2008), so using the outcome here does not bias effect estimation; see Details.
`estimand`	`"ATT"` (default) or `"ATE"`; determines the inverse-probability weights used inside the balance criteria and returned in `weights`.
`criterion`	The selection criterion for λ: `"prog"` (default) = weighted ASMD of the prognostic score (the paper's headline "Prog (Ave)"); `"smd"` = mean weighted ASMD over the covariates; `"ks"` = mean weighted Kolmogorov-Smirnov statistic over the covariates; `"logloss"` = negative Bernoulli log-likelihood of treatment assignment (the prediction-accuracy criterion in the lineage of Xie et al. 2019).
`prog.target`	Only used when `criterion = "prog"`: `"average"` (default) targets the gamma-mixed prognostic score $\bar{g}$; naming a single element of `prog.methods` (or a column of `prog.matrix`) targets that candidate prognostic score instead (the paper's "Prog (g_k)" variants).
`ps.methods`	Character vector of candidate propensity score learners. The fixed, explicit default is `c("glm", "rpart", "ranger", "xgboost")` (no auto-detection: a machine-dependent candidate set is irreproducible science). Any `"SL.*"` string is accepted verbatim and passed through to **SuperLearner**. The order defines the tie-break preference (see Details).
`prog.methods`	Character vector of candidate learners for the untreated-only prognostic models $g_k(0, X)$; same menu and default as `ps.methods`.

`ps.matrix` Optional n x M numeric matrix of user-supplied candidate propensity scores (values strictly in (0, 1), column names required). Overrides `ps.methods`; the columns are clipped like fitted candidates.

`prog.matrix` Optional n x K numeric matrix of user-supplied candidate prognostic scores (column names required). Overrides `prog.methods`. Requires `outcome` (gamma is selected by outcome-prediction MSE among untreated units).

`ps.append` Optional extra candidate propensity scores appended AFTER the candidates from `ps.methods` or `ps.matrix` (default `NULL`): a numeric vector of length n (one candidate, labeled `"append"`) or a numeric matrix / all-numeric data frame with n rows (unique, non-empty column names required, as for `ps.matrix`). Values must lie strictly in (0, 1) and each column is clipped to `clip` before averaging, exactly like every other candidate. Because appended columns come last, grid tie-breaking (first minimum) favors the base candidates (see Details).

`prog.append` Optional extra candidate prognostic scores appended AFTER the candidates from `prog.methods` or `prog.matrix` (default `NULL`); same vector/matrix/data-frame forms and naming rules as `ps.append`, but the values are unrestricted finite reals. Requires `outcome`.

`average` If `FALSE`, the λ grid is restricted to the simplex VERTICES, i.e., the best single PS candidate by `criterion` is selected (the "best single learner" variants of the paper's supplement).

`family` The prognostic-model family: `gaussian()` (default) or `binomial()` only. With `binomial()`, the gamma-selection MSE is the Brier score – the formula is unchanged; note that the paper's simulations validated continuous outcomes.

`step` The simplex-grid increment for BOTH γ and λ (default `0.05`, the paper's value). Must evenly divide 1.

`clip` Length-2 numeric: each candidate propensity score column is clipped to `[clip[1], clip[2]]` BEFORE averaging (default `c(0.01, 0.99)`, the paper's constants). The average is never re-clipped: a convex combination of values in the clipping interval stays in it.

`s.d.denom` The group whose unweighted standard deviation standardizes mean differences in the ASMD-based criteria, passed to `cobalt::col_w_smd()`: `"treated"` (default; the paper's supplement uses the unweighted TREATED-group SD for both the ATT and the ATE), `"control"`, `"pooled"`, or `"all"`.

`cv` Number of cross-validation folds `V` for the `SuperLearner::SuperLearner()` `cvControl` when `"SL.*"` learners are used (default `5`).

`control` A named list of per-learner hyperparameter overrides, e.g. `list(ranger = list(num.trees = 1000), xgboost = list(nrounds = 200))`. Entries for `"glm"` are passed to `stats::glm()`, for `"rpart"` to `rpart::rpart.control()`, for `"ranger"` to `ranger::ranger()`, and for `"xgboost"` to the `params` list of `xgboost::xgb.train()` (with `nrounds` and `verbose` recognized as top-level arguments). The resolved values are stored in `info$learners`. Multi-threaded engines run single-threaded by default (`ranger` `num.threads = 1`, `xgboost` `nthread = 1`) in line with CRAN's at-most-2-cores policy; raise these via `control` to speed up real analyses on your own machine.

`keep.fits` If `TRUE`, the fitted learner objects are retained in `fits`, enabling `predict.psave()`. Default `FALSE`.

keep.path	If TRUE (default), the full λ criterion path is stored in path. Automatically set to NULL with a message if the grid exceeds 100,000 rows.
verbose	If TRUE, progress messages report the learner set, the grid sizes, the selected λ/γ, and the criterion value.
...	Reserved for future use; supplying unused arguments triggers a warning.

Details

Algorithm:

1. **Candidate propensity scores.** Each learner in ps.methods is fit on all n units and predicts $\hat{e}_m(X_i) = P(A_i = 1 \mid X_i)$ in-sample; each column is clipped to clip before averaging. Extra user-supplied candidates given via ps.append are validated, clipped identically, and appended AFTER the base candidates (from ps.methods or ps.matrix), so the first-minimum tie-break favors the base set.
2. **Model-averaged prognostic score.** Each learner in prog.methods is fit on the **untreated units only** and predicts $\hat{g}_k(0, X_i)$ for all n units; prog.append columns are appended after these candidates. The mixing weights γ minimize the unweighted untreated-set mean squared error
$$\mathrm{MSE}(\gamma) = \mathrm{mean}_{i:A_i=0}\big(Y_i - \textstyle\sum_k \gamma_k \hat{g}_k(0, X_i)\big)^2$$
over simplex_grid(K, step); $\bar{g} = \sum_k \gamma_k \hat{g}_k$.
3. **Model-averaged propensity score.** The mixing weights λ minimize criterion over simplex_grid(M, step), where each grid row implies $\bar{e}_\lambda = \sum_m \lambda_m \hat{e}_m$ and the weights W_i below.

Inverse-probability weights (exact supplement formulas):

With $\bar{e}_i$ the averaged propensity score:

$$\mathrm{ATT}: \quad W_i = 1 \ (A_i = 1), \qquad W_i = \bar{e}_i/(1-\bar{e}_i) \ (A_i = 0);$$

$$\mathrm{ATE}: \quad W_i = 1/\bar{e}_i \ (A_i = 1), \qquad W_i = 1/(1-\bar{e}_i) \ (A_i = 0).$$

These are identical to WeightIt::get_w_from_ps() at the same estimand.

Selection criteria:

At each grid row λ:

logloss $-\mathrm{mean}\{A_i \log \bar{e}_i + (1-A_i)\log(1-\bar{e}_i)\}$ (finite by clipping).

smd the mean over covariate columns j of
$$\mathrm{ASMD}_j = \Big|\frac{\sum_{A_i=1} W_i X_{ij}}{\sum_{A_i=1} W_i} - \frac{\sum_{A_i=0} W_i X_{ij}}{\sum_{A_i=0} W_i}\Big| / s_j,$$
where s_j is the **unweighted sample SD** of X_j in the s.d.denom group (the treated group for both estimands, per the paper's supplement).

ks the mean over covariates of the proper weighted-eCDF Kolmogorov-Smirnov statistic $\sup_x |F_1^w(x) - F_0^w(x)|$ with $F_a^w(x) = \sum_{i:A_i=a} W_i 1(X_{ij} \le x) / \sum_{i:A_i=a} W_i$; for binary columns this equals the absolute difference in weighted proportions.

prog the ASMD formula applied to the single column $\bar{g}$ (prog.target = "average") or $\hat{g}_k$ (prog.target names a learner). THE DEFAULT.

When `prog.target` names a single prognostic candidate, `criterion.value` is the weighted ASMD of that candidate's prognostic score, whereas the `prog` column of the `diagnostics` table always refers to the model-averaged prognostic score $\bar{g}$; the two values therefore need not coincide, and `summary.psave()` prints a footnote to this effect. Faithful to the published method, the `smd` and `prog` criteria standardize **all** columns – including binary ones – by the plain unweighted sample SD (uniform sample-SD standardization; `bin.vars = FALSE` is passed to `cobalt::col_w_smd()` for every column). The display-oriented `balance` component and `bal.tab.psave()` use **cobalt**'s native conventions instead; see `vignette("method-details", "psAve")`.

Simplex grid and tie-breaking:

Both grids are enumerated by `simplex_grid()` in integer arithmetic (every valid grid point is present by construction) in a documented order: the first component descends from 1 to 0, then the second on the remainder, and so on – the first row puts all weight on the first candidate. Ties in any argmin take the FIRST row attaining the minimum (within a 1e-9 relative numerical tolerance), so ties favor learners listed earlier in `ps.methods`/`prog.methods`; candidates appended via `ps.append`/`prog.append` always come last and therefore lose ties to the base candidates. The tolerance is deliberate: the criterion values come from floating-point matrix algebra whose lowest-order bits can differ across BLAS implementations, so an exact bitwise `which.min()` would not be reproducible across machines, whereas the tolerant first-minimum rule is. `gamma` always minimizes the unweighted untreated-set MSE; these rules are fixed and not arguments.

Why using the outcome does not bias the design:

Prognostic models see the outcomes of **untreated units only**, and the criterion compares weighted *covariate-like summaries* (the prognostic score) between arms – it never uses a treated-untreated outcome contrast (Hansen 2008; Stuart, Lee and Leacy 2013). This is the same argument under which prognostic-score balance diagnostics are recommended for propensity score analyses; see `vignette("method-details", "psAve")`.

Reproducibility:

There is no `seed` argument: call `set.seed()` before `psave()` – stochastic learners are `ranger` and `xgboost`. `info$learners` records the resolved hyperparameters and engine package versions.

Relation to other software:

WeightIt::method_super with `SL.method = "method.balance"` (Pirracchio and Carone 2018) selects a SuperLearner combination by *covariate* balance and is weighting-only. `psave()` targets *prognostic-score* balance on an exhaustive simplex grid, and returns a score vector equally usable for matching and weighting. See psave-details and `vignette("method-details", "psAve")` for the differences from the paper's reference implementation.

Value

An object of class `"psave"`: a list with components

`ps` numeric(n), named by `rownames(data)`: the model-averaged propensity score $\bar{e}$ – the deliverable for `MatchIt::matchit(distance = )` / `WeightIt::weightit(ps = )`.

`prog` numeric(n), named: the model-averaged prognostic score $\bar{g}$; `NULL` when `outcome` was not supplied (possible only for the outcome-free criteria `"smd"`, `"ks"`, and `"logloss"`).

`lambda` named numeric(M): the selected PS mixing weights, where M counts the base candidates plus any `ps.append` columns.

`gamma` named numeric(K) (or `NULL`): the selected prognostic mixing weights, where K counts any `prog.append` columns.

`weights` numeric(n): the IPW at `estimand` implied by `ps`.

`ps.candidates` n x M matrix of clipped candidate propensity scores (fitted or user-supplied; `ps.append` columns come last).

`prog.candidates` n x K matrix of candidate prognostic predictions for all n units (or `NULL`); `prog.append` columns come last.

`criterion`, `criterion.value` the criterion used and its value at the selected λ (paper-faithful standardization).

`diagnostics` (M+1) x 4 data frame: all four criteria (`psave_criteria()`) for each candidate and for the selected average – the "was averaging worth it?" table. Its `prog` column always uses the model-averaged prognostic score $\bar{g}$, even when `prog.target` names a single candidate (see Details); it then need not equal `criterion.value`.

`path` data frame of the full λ grid (M columns + `value` + logical `selected`) in grid order, or `NULL`.

`gamma.mse` named numeric(K+1): untreated MSE per prognostic candidate and for the selected average (or `NULL`).

`balance` data frame (covariates + `prog`): unweighted and weighted SMD and KS (cobalt display conventions).

`treat` integer(n) 0/1 treatment as used.

`covs` numeric n x p balance-covariate matrix with `attr(, "bin.vars")`.

`estimand`, `s.d.denom`, `prog.target`, `average` scalars, as resolved.

`outcome.name` name of the outcome variable, or `NA`.

`formula`, `data` as supplied (they power `psave_match()`, `psave_weight()` and `print.psave()`; note the memory cost of storing `data`).

`fits` list of fitted learners iff `keep.fits = TRUE` (enables `predict.psave()`); otherwise `NULL`.

`info` list: `step`, `clip`, `cv`, `family`, `grid.size`, `n`, `learners` (labels, resolved hyperparameters, engine versions), `psAve.version`.

`call` the matched call.

References

Kabata D, Stuart EA, Shintani A (2024). Prognostic score-based model averaging approach for propensity score estimation. *BMC Medical Research Methodology*, 24, 228. doi:10.1186/s12874-02402350y

Hansen BB (2008). The prognostic analogue of the propensity score. *Biometrika*, 95(2), 481-488. doi:10.1093/biomet/asn004

Stuart EA, Lee BK, Leacy FP (2013). Prognostic score-based balance measures can be a useful diagnostic for propensity score methods in comparative effectiveness research. *Journal of Clinical Epidemiology*, 66(8), S84-S90. doi:10.1016/j.jclinepi.2013.01.013

Xie Y, Zhu Y, Cotton CA, Wu P (2019). A model averaging approach for estimating propensity scores by optimizing balance. *Statistical Methods in Medical Research*, 28(1), 84-101. doi:10.1177/0962280217715487

See Also

psave_match(), psave_weight(), simplex_grid(), psave_criteria(), bal.tab.psave(), plot.psave(), summary.psave(), predict.psave()

Examples

```
if (requireNamespace("MatchIt", quietly = TRUE)) {
  data("lalonde", package = "MatchIt")

  # Paper-headline "Prog (Ave)", ATT. For speed, this example restricts the
  # candidate learners; the default is
  # ps.methods = prog.methods = c("glm", "rpart", "ranger", "xgboost").
  meths <- if (requireNamespace("rpart", quietly = TRUE)) c("glm", "rpart") else "glm"
  fit <- psave(treat ~ age + educ + race + married + nodegree + re74 + re75,
               data = lalonde, outcome = ~ re78,
               ps.methods = meths, prog.methods = meths)
  fit
  summary(fit)

  # hand off to MatchIt (canonical explicit call):
  m <- MatchIt::matchit(treat ~ age + educ + race + married + nodegree +
                          re74 + re75,
                        data = lalonde, distance = fit$ps)
  # or, reusing the stored formula and data (no retyping hazard):
  m2 <- psave_match(fit)
}
```

---

psave-details *Method details: relation to the paper's reference implementation*

Description

psAve implements the published method of Kabata, Stuart and Shintani (2024) rather than reproducing its reference code line by line. Five documented defects/quirks of the reference implementation are deliberately **fixed** (each fix follows the paper's *stated* definitions):

1. **Integer simplex grid.** The reference code enumerated the mixing-weight grid with `expand.grid()` and kept rows passing an exact floating-point `rowSums(gr) == 1` test, silently dropping about 10.6% of the valid grid points for M = 4 candidates at step 0.05 (1,584 of 1,771 kept). simplex_grid() enumerates integer compositions, so every valid point is present by construction, and the enumeration order makes the first-minimum tie-break a reproducible rule.
2. **Proper weighted-eCDF KS statistic.** The reference `Fks` computed `ks.test()` on covariate values *multiplied by* the weights, which is not the paper's weighted-eCDF definition. **psAve** computes $\sup_x |F_1^w(x) - F_0^w(x)|$ with weighted empirical CDFs in each arm (as cobalt::col_w_ks() does).
3. **Proper binomial family.** The reference code fit binary-response SuperLearner models with `gaussian(link = "logit")`; **psAve** uses `binomial()` throughout for treatment models.

4. **No per-set `scale()`.** The reference code standardized fitting and prediction sets *separately*, an inconsistent transformation. **psAve** passes raw covariates to all engines.
5. **Strict complete-case alignment.** The reference `Fasmd` applied `na.omit()` to a covariate while using full-length treatment and weight vectors, silently misaligning rows in the presence of missing data. **psAve** refuses missing values in any used variable (error, never drop), and names all returned score vectors by `rownames(data)`.

Details

Clipping:

Candidate propensity scores are clipped to `clip` (default [0.01, 0.99], the paper's constants) **before** averaging. The average is never re-clipped: a convex combination of values inside the clipping interval cannot leave it.

Criterion vs. display conventions:

The `smd`/`prog` *selection criteria* standardize all columns (including binary ones) by the plain unweighted sample SD of the `s.d.denom` group – the paper's convention (its reference `Fasmd` uses `sd()`), implemented by passing `bin.vars = FALSE` for every column to `cobalt::col_w_smd()`. The display-oriented `balance` component and `bal.tab.psave()` follow **cobalt**'s native conventions (binary columns use $\sqrt{p(1-p)}$). For `criterion = "prog"` the denominator is a positive constant across the λ grid, so the selected λ is invariant to this choice; the *reported* `criterion.value` uses the paper's convention. For the KS criterion the two conventions coincide on binary columns.

Relation to other software:

The closest existing functionality is WeightIt::method_super with `SL.method = "method.balance"` (Pirracchio and Carone 2018): a covariate-balance-targeted SuperLearner for *weighting only*. **psAve** differs in targeting **prognostic-score** balance (Hansen 2008; Stuart, Lee and Leacy 2013), in searching an exhaustive simplex grid with a documented tie-break instead of a convex-optimization meta-learner, and in returning a plain score vector equally usable for **matching** (`MatchIt::matchit(distance = )`) and weighting. No other package implements propensity score model averaging in the lineage of Xie et al. (2019).

Reproducibility:

There is no `seed` argument (setting the global RNG inside a function is an R anti-pattern): call `set.seed()` before `psave()` when stochastic learners (`ranger`, `xgboost`) are among the candidates. The resolved hyperparameters and engine package versions are recorded in `fit$info$learners`.

References

Kabata D, Stuart EA, Shintani A (2024). Prognostic score-based model averaging approach for propensity score estimation. *BMC Medical Research Methodology*, 24, 228. doi:10.1186/s12874-02402350y

Hansen BB (2008). The prognostic analogue of the propensity score. *Biometrika*, 95(2), 481-488. doi:10.1093/biomet/asn004

Stuart EA, Lee BK, Leacy FP (2013). Prognostic score-based balance measures can be a useful diagnostic for propensity score methods in comparative effectiveness research. *Journal of Clinical Epidemiology*, 66(8), S84-S90. doi:10.1016/j.jclinepi.2013.01.013

Xie Y, Zhu Y, Cotton CA, Wu P (2019). A model averaging approach for estimating propensity scores by optimizing balance. *Statistical Methods in Medical Research*, 28(1), 84-101. doi:10.1177/0962280217715487

Pirracchio R, Carone M (2018). The Balance Super Learner: A robust adaptation of the Super Learner to improve estimation of the average treatment effect in the treated based on propensity score matching. *Statistical Methods in Medical Research*, 27(8), 2504-2518. doi:10.1177/0962280216682055

See Also

`psave()`, `simplex_grid()`, `psave_criteria()`, `vignette("method-details", "psAve")`

`psave_criteria` *Compute all four psAve selection criteria for a propensity score vector*

Description

Evaluates, for an arbitrary propensity score vector, the four selection criteria used by `psave()`: the treatment-assignment log loss, the mean weighted absolute standardized mean difference (ASMD) of the covariates, the mean weighted Kolmogorov-Smirnov (KS) statistic of the covariates, and the weighted ASMD of a prognostic score. This function powers the `diagnostics` table of a `psave` object (the "was averaging worth it?" comparison) and is exported as a methods-research utility.

Usage

```
psave_criteria(
  ps,
  treat,
  covs,
  prog = NULL,
  estimand = c("ATT", "ATE"),
  s.d.denom = "treated",
  bin.vars = NULL
)
```

Arguments

`ps`	Numeric vector of propensity scores, strictly inside (0, 1).
`treat`	Treatment vector; coerced to 0/1 like the left-hand side of the `psave()` formula (numeric 0/1, logical, or two-level factor/character with the second level treated).
`covs`	Numeric matrix (or all-numeric data frame) of covariates, one row per unit. Factors must already be expanded to dummy columns (as in the `covs` component of a `psave` object).
`prog`	Optional numeric vector: a prognostic score. If `NULL`, the `prog` criterion is returned as `NA`.

`estimand` "ATT" (default) or "ATE"; determines the weights (see Details).

`s.d.denom` Group whose (unweighted) standard deviation standardizes the mean differences: "treated" (default; the paper's convention for BOTH estimands), "control", "pooled", or "all".

`bin.vars` Optional logical vector flagging binary columns of covs, used for the KS criterion only; if NULL, columns with exactly two distinct values are detected automatically. The smd and prog criteria always use uniform sample-SD standardization (bin.vars = FALSE for all columns), which is the convention of the published method (see Details).

Details

Weights are the inverse-probability weights implied by ps at estimand: for the ATT, $W_i = 1$ for treated units and $e_i/(1-e_i)$ for untreated units; for the ATE, $1/e_i$ and $1/(1-e_i)$. The four criteria are:

`logloss` $-\text{mean}\{A_i \log e_i + (1-A_i)\log(1-e_i)\}$.

`smd` the mean over covariates j of $|\bar{X}^w_{1j} - \bar{X}^w_{0j}|/s_j$, where $\bar{X}^w_{aj}$ is the weighted mean of X_j in arm a and s_j is the unweighted sample SD of X_j in the s.d.denom group. Computed via `cobalt::col_w_smd()`.

`ks` the mean over covariates of the proper weighted-eCDF KS statistic $\sup_x |F^w_1(x) - F^w_0(x)|$, computed via `cobalt::col_w_ks()`; for binary columns this is the absolute difference in weighted proportions.

`prog` the same weighted ASMD formula applied to the single column prog.

Faithful to the published method (and its reference implementation), the smd and prog criteria standardize **every** column, including binary ones, by the plain unweighted sample SD (sd(), the $n-1$ formula) of the s.d.denom group – i.e., bin.vars = FALSE is passed to `cobalt::col_w_smd()` for all columns. cobalt's own display convention (binary columns standardized by $\sqrt{p(1-p)}$) is used only in the display-oriented balance component of a psave object and in `bal.tab.psave()`. For the KS criterion the two conventions coincide.

Value

A named numeric vector with elements logloss, smd, ks, and prog (the last is NA when prog = NULL).

References

Kabata D, Stuart EA, Shintani A (2024). Prognostic score-based model averaging approach for propensity score estimation. *BMC Medical Research Methodology*, 24, 228. doi:10.1186/s12874-02402350y

Hansen BB (2008). The prognostic analogue of the propensity score. *Biometrika*, 95(2), 481-488. doi:10.1093/biomet/asn004

Xie Y, Zhu Y, Cotton CA, Wu P (2019). A model averaging approach for estimating propensity scores by optimizing balance. *Statistical Methods in Medical Research*, 28(1), 84-101. doi:10.1177/0962280217715487

See Also

psave(), simplex_grid(), cobalt::col_w_smd(), cobalt::col_w_ks()

Examples

```
set.seed(1)
n <- 200
x1 <- rnorm(n); x2 <- rbinom(n, 1, 0.4)
a <- rbinom(n, 1, plogis(-0.5 + x1 + 0.5 * x2))
ps <- pmin(pmax(fitted(glm(a ~ x1 + x2, family = binomial())), 0.01), 0.99)
g <- 1 + 0.5 * x1 - 0.2 * x2   # a (toy) prognostic score
psave_criteria(ps, a, cbind(x1 = x1, x2 = x2), prog = g, estimand = "ATT")
```

---

psave_match *Match on a model-averaged propensity score*

Description

Convenience pass-through to MatchIt::matchit(): matches on the model-averaged propensity score of a psave() fit, reusing the formula and data stored in the object. Equivalent to the canonical explicit call

```
MatchIt::matchit(<formula>, data = <data>, distance = fit$ps, ...)
```

but with no opportunity for row misalignment between the two steps. All ... arguments are forwarded verbatim; the return value is an ordinary matchit object, so the full **MatchIt/cobalt** toolkit applies.

Usage

```
psave_match(x, ...)
```

Arguments

x	A psave object.
...	Arguments forwarded verbatim to MatchIt::matchit() (e.g., method, caliper, ratio, replace).

Value

A matchit object; see MatchIt::matchit().

See Also

psave(), psave_weight(), MatchIt::matchit()

Examples

```
data("lalonde", package = "MatchIt")
fit <- psave(treat ~ age + educ + married + re74, data = lalonde,
             outcome = ~ re78, ps.methods = "glm", prog.methods = "glm")
m <- psave_match(fit, method = "nearest", caliper = 0.2)
cobalt::bal.tab(m, distance = data.frame(prog = fit$prog))
```

`psave_weight` *Weight by a model-averaged propensity score*

Description

Convenience pass-through to `WeightIt::weightit()`: constructs balancing weights from the model-averaged propensity score of a `psave()` fit at the fitted estimand, reusing the stored formula and data. Equivalent to the canonical explicit call

`WeightIt::weightit(<formula>, data = <data>, ps = fit$ps, estimand = fit$estimand, ...)`

All `...` arguments are forwarded verbatim; the return value is an ordinary `weightit` object.

Usage

```
psave_weight(x, ...)
```

Arguments

`x`	A psave object.
`...`	Arguments forwarded verbatim to `WeightIt::weightit()`.

Value

A `weightit` object; see `WeightIt::weightit()`.

See Also

`psave()`, `psave_match()`, `WeightIt::weightit()`, `WeightIt::get_w_from_ps()`

Examples

```
data("lalonde", package = "MatchIt")
fit <- psave(treat ~ age + educ + married + re74, data = lalonde,
             outcome = ~ re78, ps.methods = "glm", prog.methods = "glm")
w <- psave_weight(fit)
cobalt::bal.tab(w, distance = data.frame(prog = fit$prog))
```

`simplex_grid` *Enumerate the discretized probability simplex*

Description

Enumerates all points of the probability simplex $\{\lambda \in [0,1]^M : \sum_m \lambda_m = 1\}$ on a regular grid with increment `step`, using **integer arithmetic**: with $S = 1/\texttt{step}$ steps, every integer composition $(c_1, \ldots, c_M)$ with $\sum_m c_m = S$ and $c_m \geq 0$ is listed and returned as c/S. This is the grid over which `psave()` searches for the propensity score mixing weights λ and the prognostic mixing weights γ.

Usage

```
simplex_grid(M, step = 0.05)
```

Arguments

`M`	Integer; the number of mixture components (grid columns).
`step`	Numeric; the grid increment. Must evenly divide 1 (checked in integer arithmetic: with `n_steps = round(1/step)`, the call errors unless `abs(n_steps * step - 1) < 1e-8`). Default `0.05`, the value used in Kabata, Stuart and Shintani (2024).

Details

The number of grid points is exactly $\binom{S+M-1}{M-1}$; e.g., `M = 4`, `step = 0.05` gives `choose(23, 3) = 1771` points. Because the grid is built from integer compositions, every valid point is present by construction; the reference implementation of the paper instead filtered `expand.grid()` rows with a floating-point `rowSums(gr) == 1` test, which silently dropped about 10.6% of the valid points for `M = 4`, `step = 0.05`.

Enumeration order (the tie-breaking rule). Rows are generated by recursive descent: c_1 runs from S down to 0; within each value of c_1, c_2 runs from the remainder down to 0; and so on. The first row is therefore $(1, 0, \ldots, 0)$ and the last row is $(0, \ldots, 0, 1)$. All grid searches in `psave()` resolve ties by taking the **first** row attaining the minimum, within a 1e-9 relative tolerance of the minimum, so ties favor learners listed earlier in `ps.methods` / `prog.methods`. The tolerance is deliberate: criterion values are computed with floating-point matrix algebra whose lowest-order bits can differ across BLAS implementations, so an exact bitwise `which.min()` would not be reproducible across machines, whereas the tolerant first-minimum rule is.

Value

A numeric matrix with $\binom{S+M-1}{M-1}$ rows and `M` columns; each row sums to 1 exactly (in integer arithmetic before the single final division by S).

References

Kabata D, Stuart EA, Shintani A (2024). Prognostic score-based model averaging approach for propensity score estimation. *BMC Medical Research Methodology*, 24, 228. doi:10.1186/s12874-02402350y

See Also

`psave()`, `psave_criteria()`

Examples

```
simplex_grid(2, step = 0.25)
nrow(simplex_grid(4, step = 0.05))  # choose(23, 3) = 1771

# first row = all weight on the first component; last = on the last:
head(simplex_grid(3, step = 0.25), 3)
tail(simplex_grid(3, step = 0.25), 3)
```

`summary.psave` *Summarize a psave object*

Description

Produces (a) the selected mixing-weight tables λ and γ, (b) the `diagnostics` table (all four selection criteria for every candidate propensity score and for the selected average – the "was averaging worth it?" comparison), and (c) the full balance table (all covariates plus the prognostic score; unweighted vs. weighted SMD and KS, with a $*$ marker at weighted SMD > 0.1).

Usage

```
## S3 method for class 'psave'
summary(object, un = TRUE, candidates = TRUE, ...)

## S3 method for class 'summary.psave'
print(x, digits = 3, ...)
```

Arguments

`object`	A `psave` object.
`un`	If `TRUE` (default), the balance table includes the unweighted columns.
`candidates`	If `TRUE` (default), the per-candidate `diagnostics` table is included.
`...`	Ignored.
`x`	A `summary.psave` object.
`digits`	Number of significant digits to print. Default 3.

Value

For `summary.psave()`, an object of class `"summary.psave"`: a list with elements `lambda`, `gamma`, `gamma.mse`, `diagnostics`, `balance`, `criterion`, `criterion.value`, `prog.target`, `estimand`, `average`, `nn`, and `call`. `print.summary.psave()` returns `x` invisibly.

See Also

`psave()`, `print.psave()`, `bal.tab.psave()`

Index