Package 'fastPLS'

October 20, 2026

Type Package

Title Fast Partial Least Squares for High-Dimensional Data

Version 0.4

Date 2026-10-15

Description Fast implementations of partial least squares models for high-dimensional regression and classification. The 'fastPLS' software provides compiled implementations of PLS-SVD, a SIMPLS-family estimator, OPLS and kernel PLS, together with truncated singular value decomposition backends, discriminant classifiers, cross-validation utilities and optional 'CUDA' or Apple 'Metal' acceleration when the required system libraries are available. Compact latent prediction and memory-aware numerical routes support analyses with large predictor or multivariate-response matrices.

License MIT + file LICENSE

URL https://github.com/tkcaccia/fastPLS

BugReports https://github.com/tkcaccia/fastPLS/issues

biocViews Software, DimensionReduction, Classification, Regression, GenePrediction, GeneExpression, Metabolomics, SingleCell, GPU

SystemRequirements Optional OpenBLAS development libraries for faster CPU matrix operations; optional NVIDIA CUDA Toolkit with CUDA Runtime, cuBLAS, cuSOLVER and cuRAND plus a compatible separately managed NVIDIA driver; optional Apple Metal framework on macOS. CPU-only builds do not require GPU software.

Depends R (>= 4.6.0)

Imports methods, float, gpuinfo (>= 0.2.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

NeedsCompilation yes

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.1.0

Author Stefano Cacciatore [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7052-7156>),
Dupe Ojo [aut] (ORCID: <https://orcid.org/0000-0002-5301-8592>),
Leonardo Tenori [aut] (ORCID: <https://orcid.org/0000-0001-6438-059X>),
Alessia Vignoli [aut] (ORCID: <https://orcid.org/0000-0003-4038-6596>)

Maintainer Stefano Cacciatore <tkcaccia@gmail.com>

Repository CRAN

Date/Publication 2026-10-20 12:20:09 UTC

Contents

cuda_info 2
evaluate 3
fastcor 5
fastPLS_blas 6
fastsvd 7
plot.fastPLS 9
plot.permutation 10
pls 11
pls.double.cv 15
pls.single.cv 20
predict.fastPLS 25
ViP 26

Index **28**

`cuda_info` *Report CUDA Build and Runtime Capability*

Description

Distinguish a functional CUDA build from a package built without CUDA and from an explicit diagnostic-only build.

Usage

```
cuda_info()
```

Details

A functional result requires both CUDA code compiled into fastPLS and at least one device reported by the CUDA runtime. fastPLS never substitutes the CPU backend for an explicit CUDA request. CUDA version integers use the CUDA Runtime API representation; for example, 13000 denotes CUDA 13.0.

Hardware and shared-library diagnostics come from **gpuinfo**. `hardware_available` reports its independent CUDA driver probe and does not override the runtime result from the libraries linked into fastPLS. The `libraries` table includes `required_by_fastPLS` to identify the CUDA driver, runtime, cuBLAS, cuSOLVER, and cuRAND. cuFFT and NCCL are not required. A missing separately discoverable library does not disable a working fastPLS binary, which can use linked or bundled libraries.

Value

A named list containing `status`, `compiled`, `available`, `diagnostic_only`, `device_count`, `runtime_version`, `driver_version`, `no_cpu_fallback`, and `hardware_available`, and a `libraries` diagnostic table. `available` and `status` describe this fastPLS build and its own CUDA runtime probe.

See Also

`gpuinfo::cuda_info()`, `gpuinfo::cuda_libraries()`, `pls`

Examples

```
fastPLS::cuda_info()
```

---

evaluate *Evaluate prediction performance*

Description

Computes common classification or regression performance metrics from observed and predicted values. The function accepts vectors, matrices, classification score or rank matrices, or complete fastPLS prediction results.

Usage

```
evaluate(
    observed,
    predicted,
    ytrain = NULL,
    bycol = TRUE,
    relative_epsilon = .Machine$double.eps,
    na.rm = TRUE
)
```

Arguments

observed	Observed response values. Use a factor or character vector for classification, a numeric vector or matrix for regression, or a one-hot matrix for classification.
predicted	Predicted values. Use a factor or character vector for predicted classes, a numeric vector or matrix for regression, a class-score or ranked-label matrix for classification, or the complete object returned by `predict()` for a fitted fastPLS model.
ytrain	Optional training response for independent-test regression Q2. When supplied, each response is centered on its corresponding training mean before denominator sums are aggregated across responses. When omitted, Q2 is returned as `NA` rather than being silently equated with R2.
bycol	For multivariate regression, calculate and return metrics for each response column. The default is `TRUE` for direct `evaluate()` calls.
relative_epsilon	Values with absolute observed response below this threshold are ignored for relative-error metrics.
na.rm	Remove incomplete observations before computing metrics.

Details

For classification, the returned metrics include accuracy, the no-information rate, lift accuracy, balanced accuracy, macro precision, macro recall, macro F1, Cohen's kappa, and a confusion matrix. The no-information rate is the accuracy obtained by always predicting the most prevalent observed class. Lift accuracy is accuracy divided by this baseline; values above one indicate an improvement over the majority-class baseline. When class scores or ranked labels are supplied, top-k accuracy is inferred and reported automatically.

For regression, the returned metrics include R2, Q2, RMSD/RMSE, MAE, bias, median relative error percentage, mean absolute percentage error, ratio of performance to deviation, Pearson correlation, and Spearman correlation. These include the main spectral prediction metrics used by Vignoli et al. (2025): median relative error percentage, RMSE, R2, and RPD. Set `bycol = FALSE` to return only the aggregate metrics and avoid calculating a separate metric vector for every response column. For multivariate responses, aggregate R2 centers each response column on its own observed mean before summing the residual and reference sums of squares. Independent-test Q2 analogously centers each response column on the corresponding training mean supplied through `ytrain`. If that reference is unavailable, Q2 is `NA` and `metric_definitions` records why it was not calculated.

Value

A list with `task`, `metrics`, `metric_definitions`, and optionally `per_response`, `per_class`, `confusion`, and `topk`. Top-k ranks are inferred from score or ranked-prediction columns. For a prediction object with several component counts, `metrics` has one row per count and `by_component` contains each complete evaluation. A `notes` element is included only when the evaluation has an explanatory note to report.

Examples

```
evaluate(iris$Species, iris$Species)

set.seed(1)
y <- mtcars$mpg
pred <- y + rnorm(length(y), sd = 2)
evaluate(y, pred)$metrics
```

`fastcor` *Fast Pearson Correlation*

Description

Centers and normalizes rows (or columns when `byrow = FALSE`) and then computes Pearson correlations using fast matrix cross-products. This function does not rank-transform the inputs and therefore does not compute Spearman correlation.

Usage

```
fastcor(a, b = NULL, byrow = TRUE, diag = TRUE, n.cores = NULL)
```

Arguments

`a`	Numeric matrix.
`b`	Optional numeric matrix with the same row or column orientation as `a`.
`byrow`	Logical; when `TRUE`, rows are correlated. When `FALSE`, columns are correlated.
`diag`	Logical; when `b` is supplied and `diag = TRUE`, return only the pairwise diagonal correlations between matching rows (or columns).
`n.cores`	Number of CPU cores requested for the compiled matrix operations. An explicit value takes precedence over `options(n.cores = ...)`.

Value

A correlation matrix, or a numeric vector of diagonal correlations when `b` is supplied with `diag = TRUE`.

Author(s)

Stefano Cacciatore, Leonardo Tenori, Dupe Ojo, Alessia Vignoli

See Also

`pls.single.cv`, `pls.double.cv`

Examples

```
data(iris)
x <- as.matrix(iris[1:10, -5])
fastcor(x)
```

`fastPLS_blas` *Report the numerical library selected when fastPLS was compiled*

Description

This reports the CPU matrix library selected by the package configuration, rather than attempting to infer a library from the current R session. macOS builds normally report `"Accelerate"`. Linux and Windows builds report `"OpenBLAS"` when OpenBLAS was found or explicitly requested, and `"R BLAS/LAPACK"` when the package used R's portable fallback.

Usage

```
fastPLS_blas(details = TRUE)
```

Arguments

`details`	Logical. Return a detailed named list when `TRUE`, or the former scalar backend name when `FALSE`.

Details

With `details = TRUE`, the result additionally reports the library version, configuration string, selected CPU core, parallel runtime, active thread count, and resolved library path when these are exposed by the linked library. OpenBLAS provides all fields except that a statically linked Windows build may not expose a separate library path. Accelerate and R BLAS/LAPACK do not expose the same runtime metadata, so unavailable fields are `NA`.

For reproducible performance benchmarks on Linux or Windows, install with `FASTPLS_USE_OPENBLAS=1` and verify both `backend` and the detailed version and core fields before running the analysis.

Value

With `details = TRUE`, a named list containing `backend`, `version`, `configuration`, `core`, `parallel`, `threads`, and `library`. With `details = FALSE`, a single character string: `"Accelerate"`, `"OpenBLAS"`, or `"R BLAS/LAPACK"`.

Examples

```
fastPLS_blas()
fastPLS_blas(details = FALSE)
```

`fastsvd` *Native Randomized Singular Value Decomposition*

Description

Computes a truncated randomized singular value decomposition (rSVD) with the CPU backend. Ordinary R numeric matrices use float64 calculations; `float::float32` matrices automatically use the native float32 path without conversion to float64.

Usage

```
fastsvd(
    x,
    nu = NULL,
    nv = NULL,
    ncomp = NULL,
    backend = NULL,
    n.cores = NULL,
    oversample = 32L,
    power = 5L,
    seed = 1L
)
```

Arguments

`x`	Dense matrix to decompose. A base R numeric matrix is processed in float64. A `float::float32` matrix is processed end to end in float32 and returns float32 left and right singular vectors. Sparse matrices should be converted by the caller.
`nu`	Number of left singular vectors to return. If `NULL`, the function uses the largest feasible rank implied by the matrix dimensions. When `ncomp` is supplied, `ncomp` controls the decomposition rank and `nu` controls only how many left vectors are kept in the returned object.
`nv`	Number of right singular vectors to return. If `NULL`, the function uses the largest feasible rank implied by the matrix dimensions. When `ncomp` is supplied, `ncomp` controls the decomposition rank and `nv` controls only how many right vectors are kept in the returned object.
`ncomp`	Optional truncated rank. When supplied, it overrides the rank implied by `nu` and `nv`; the final rank is always capped at `min(nrow(x), ncol(x))`.
`backend`	Compute backend. `cpu` runs on the host CPU. Standalone CUDA and Metal rSVD are unavailable because their reduced QR/SVD stages are not fully device-native for every matrix shape; both accelerators remain available for supported resident PLS fits. When omitted, the function uses `options(backend = ...)`, then `FASTPLS_BACKEND`, then CPU. An unavailable or unsupported accelerator selection raises an error; no CPU fallback is performed.

n.cores	Number of CPU cores requested for supported BLAS/OpenMP host operations. An explicit value takes precedence over options(n.cores = ...). The setting does not control CUDA or Metal device parallelism.
oversample	Non-negative oversampling dimension used by randomized SVD. The sketch dimension is approximately ncomp + oversample, capped by the matrix rank. Larger values can improve approximation accuracy at the cost of extra time and memory. The default starting value is 32. Panel agreement is not a guarantee for a new matrix; CPU float32 and float64 fits apply the native case-specific audit described in Details.
power	Number of randomized-SVD power iterations. The default is five on CPU. CPU float64 and float32 execution applies case-specific auditing and recovery. Each additional iteration adds matrix multiplications.
seed	Random seed used to generate the Gaussian sketch. It affects rsvd results.

Details

Randomized SVD is stochastic. CPU float32 and float64 execution audits each decomposition using normalized left/right singular-triplet residuals and an omitted-direction search. An estimated omitted-to-retained boundary ratio above 0.95 is treated as weak separation, not by itself as a failed approximation. A residual above 0.01 or an omitted-direction ratio above 1.01 triggers stronger sketches; independent-sketch agreement supplies an additional acceptance check. If the ordinary CPU attempts fail, native operator rSVD makes up to four further attempts with wider sketches and additional power iterations. Each recovery result is rechecked. If recovery exceeds its estimated 256 MiB major-workspace budget or still fails its checks, an error is returned. This budget excludes input matrices, model storage, and runtime overhead; it is not a bound on process RSS. Small CPU inputs can use a separately recorded dense-SVD route. A full-width sketch can also use a dense SVD as part of the native algorithm. Accelerator routes report their structural diagnostics and exact controls. Confirm coefficient or subspace interpretation against an independent dense decomposition when feasible, and compare seeds when conclusions are near a decision boundary.

Value

A list compatible with base::svd() containing d, u, and v, plus backend metadata backend, method, svd.method, elapsed, ncomp, and precision. For base R numeric input, u and v are float64 matrices. For float::float32 input, u and v remain float32 matrices. Both CPU precisions receive the native case-specific rSVD audit, strengthened retries, and effective controls. The float32 audit remains in single precision, and the R layer does not convert the input to float64 solely for diagnostics. Float64 input additionally reports selected singular-triplet residuals and left/right orthogonality errors. Recovery after failed CPU sketches uses strengthened native-rSVD attempts. These fields describe the numerical checks applied to the current decomposition.

Examples

```
set.seed(1)
x <- matrix(rnorm(12 * 5), 12, 5)
s64 <- fastsvd(x, ncomp = 2, backend = "cpu", seed = 1)
s64$d

x32 <- float::fl(x)
```

```
s32 <- fastsvd(x32, ncomp = 2, backend = "cpu", seed = 1)
s32$d
```

`plot.fastPLS` *Plot PLS latent scores*

Description

Draws a two-component score plot for a fitted `fastPLS` object. Optional ellipses are computed either as data confidence ellipses or Hotelling T2 score ellipses. Axis labels include predictor-space variance explained when it was requested during model fitting.

Usage

```
## S3 method for class 'fastPLS'
plot(
  x,
  comps = c(1L, 2L),
  groups = NULL,
  score.set = c("auto", "train", "test"),
  ellipse = FALSE,
  ellipse.type = c("confidence", "hotelling"),
  conf = 0.95,
  ...
)
```

Arguments

`x`	A fitted `fastPLS` object.
`comps`	Two component indices.
`groups`	Optional grouping vector used for point fills and grouped ellipses.
`score.set`	Plot `train` scores, `test` scores, or use `auto` to select stored training scores before test scores.
`ellipse`	Draw confidence ellipses when `TRUE`.
`ellipse.type`	Use `confidence` or `hotelling` ellipses.
`conf`	Confidence level.
`...`	Additional arguments passed to `plot()`.

Value

Invisibly returns the plotted score matrix.

Examples

```
X <- as.matrix(iris[, seq_len(4)])
fit <- pls(X, iris$Species,
    ncomp = 2, fit = TRUE,
    return_variance = TRUE, seed = 1
)
plot(fit, groups = iris$Species, ellipse = TRUE)
```

`plot.permutation` *Plot PLS permutation-test R2 and Q2 values*

Description

Draws the permutation-test diagnostic plot produced by `pls(..., perm.test = TRUE)`. The x-axis is the correlation between the original and permuted response structure; the y-axis is the observed or permuted R2/Q2 value. R2 is shown in blue and Q2 in red.

Usage

```
## S3 method for class 'permutation'
plot(
  x,
  ncomp = NULL,
  main = NULL,
  xlab = "Cor",
  ylab = "Value",
  col = c(R2 = "#3155B7", Q2 = "#E5332A"),
  pch = c(R2 = 16, Q2 = 15),
  legend_position = "bottomright",
  ...
)
```

Arguments

`x`	A `fastPLS` model fitted with `perm.test = TRUE`, or a permutation data frame stored in `model$permutation`.
`ncomp`	Component count to plot. Defaults to the largest component stored in the permutation table.
`main`, `xlab`, `ylab`	Plot title and axis labels.
`col`, `pch`	Colors and point symbols for R2 and Q2.
`legend_position`	Legend position passed to `legend()`.
`...`	Additional graphical parameters passed to `plot()`.

Value

Invisibly returns the plotted permutation data.

Examples

```
set.seed(1)
X <- as.matrix(iris[, seq_len(4)])
y <- iris$Sepal.Length
idx <- sample(seq_len(nrow(X)), 30)
fit <- pls(X[idx, ], y[idx], X[idx, ], y[idx],
    ncomp = 2, perm.test = TRUE, times = 5
)
plot.permutation(fit)
```

`pls` *Partial Least Squares with selectable model family and backend*

Description

Fits PLS-SVD, a SIMPLS-family estimator, OPLS, or kernel PLS models for regression or classification using a selected CPU, CUDA, or operation-split Apple Metal backend. The fitted model can include predictions for held-out samples, latent scores, fitted values, variance summaries, and optional classification heads.

Usage

```
pls(
  Xtrain,
  Ytrain,
  Xtest = NULL,
  Ytest = NULL,
  ncomp = 2,
  scaling = c("centering", "autoscaling", "none"),
  method = c("simpls", "plssvd", "opls", "kernelpls"),
  classifier = c("lda", "argmax"),
  fit = FALSE,
  bycol = FALSE,
  return_variance = TRUE,
  return_loadings = FALSE,
  proj = FALSE,
  perm.test = FALSE,
  times = 100,
  backend = NULL,
  n.cores = NULL,
  north = 1L,
  kernel = c("linear", "rbf", "poly"),
  gamma = NULL,
  degree = 3L,
  coef0 = 1,
  ...
)
```

Arguments

Xtrain	Numeric training predictor matrix or a float::float32 predictor matrix for the supported float32 route.
Ytrain	Training response. Use a numeric vector/matrix for regression or factor/character class labels for classification.
Xtest	Optional test predictor matrix.
Ytest	Optional test response for independent-test Q2Y, whose denominator uses the training-response mean. Classification labels may contain classes absent from the training data; such classes cannot be predicted and count as classification errors.
ncomp	Positive integer component count or vector of counts. Repeated values are removed. When a direct PLS-LDA fit reaches its numerical rank, all requested positions are retained and positions beyond that rank repeat the last estimable prediction and discriminant scores. The fitted object reports both requested and effective component counts.
scaling	One of centering, autoscaling, or none.
method	One of simpls, plssvd, opls, or kernelpls. simpls uses the fastPLS accelerated SIMPLS-family core. Bounded-block execution is approximate and is not classical de Jong SIMPLS.
classifier	Classification decision rule. The default lda fits a regularized linear discriminant analysis classifier on the PLS latent scores. argmax selects the class with the largest PLS-DA response score.
fit	Return fitted values, training scores, and R2Y when TRUE. The default FALSE keeps only the compact state required for prediction and avoids materializing training-only outputs.
bycol	For matrix-valued regression responses, calculate response-wise metrics in metrics. The default FALSE returns only aggregate metrics.
return_variance	Compute predictor-space latent-variable variance explained. Set to FALSE for timing/memory benchmarks that do not need plotting variance metadata.
return_loadings	Compute and store predictor loadings P. The default is FALSE because most prediction workflows only need the projection weights and response-side coefficients.
proj	Return projected Ttest when TRUE.
perm.test	Run a single-split permutation test when Xtest and Ytest are supplied. Because pls() has no grouping argument, the exchangeability unit is one training row. Training rows are permuted, the model is refitted with the same randomized-solver seed, and the permuted test-set Q2Y path is compared with the observed path.
times	Number of requested permutations. For each component, pls() returns (b + 1) / (B + 1), where b is the number of successful null fits with Q2Y at least as large as observed and B is the number of successful null fits. Failed fits are excluded from B and reported.

backend	Implementation backend: cpu for compiled CPU, cuda for CUDA-native fitting, or metal for operation-split float32 CPU/Metal fitting on Apple silicon. When omitted, options(backend = ...) defines the session default. Selecting unavailable CUDA or Metal support raises an error without a CPU fallback.
n.cores	Number of CPU cores requested for supported BLAS/OpenMP host operations. An explicit value takes precedence over options(n.cores = ...). CUDA and Metal device parallelism is managed by their native runtimes; n.cores applies only to their host-side stages.
north	Number of orthogonal components removed by OPLS. The predictive count ncomp must fit within the predictor rank remaining after filtering. Direct PLS-LDA fits retain larger requested positions by repeating the last estimable result; other OPLS routes reject an unavailable predictive direction. Orthogonal components are not counted as predictive components.
kernel	Kernel type for kernel PLS: linear, rbf, or poly.
gamma	Kernel scale. Defaults internally to 1 / ncol(Xtrain).
degree	Polynomial kernel degree.
coef0	Polynomial kernel offset.
...	Optional SVD tuning controls forwarded to the selected backend. Use the same compact names documented in fastsvd(), such as oversample, power, and seed.

Details

The CPU backend uses Apple Accelerate on macOS, prefers OpenBLAS when available on Linux and Windows, and otherwise uses the numerical libraries supplied by R. Use fastPLS_blas() to report the library chosen at compilation. options(n.cores = n) requests CPU threads when supported by that library; additional threads are not guaranteed to improve every fit.

Base R numeric matrices use float64. Supplying float::float32 predictors or numeric responses requests float32 execution without silent promotion. CUDA supports float32 and float64. The Apple Metal route requires float32 and divides work between CPU code and persistent Metal workspaces. Unsupported backend and precision combinations stop rather than silently falling back to CPU.

method = "simpls" selects the fastPLS SIMPLS-family estimator. Its component-wise route retains the classical sequential orthogonalization and deflation structure. An eligible route may instead consume a bounded block of candidates computed from one deflated state; this is an approximate SIMPLS-family estimator, not classical de Jong SIMPLS. Classification can use response-score argmax or LDA on latent scores. PLS-SVD classification cannot return more than one fewer component than the number of response classes. The requested model family is never silently replaced by another family.

All public PLS routes use randomized SVD. It is an approximate solver, and diagnostics records the effective controls and structural checks for the fitted route. Compare repeated seeds or an independent high-accuracy fit when results are close to a decision boundary or the data are strongly ill-conditioned. Explicit oversample, power, and seed values supplied through ... override the automatic controls.

LDA uses a regularized pooled within-class covariance calculation with Cholesky solves. Regularization increases through a fixed internal sequence only when factorization fails and is not user-tuned.

Value

A fastPLS object. The object is a list whose fields depend on the selected method, backend, classifier, and whether test data or optional summaries were requested. metrics is a list of complete evaluate() results, organized as metrics$fitted and metrics$test, with one element per requested component count. Common fields are:

- P: predictor loadings, with one column per latent component when return_loadings = TRUE; otherwise an empty matrix is returned.
- Q: response loadings or response-side latent coefficients.
- R: predictor weights/rotations used to project new samples into the PLS latent space.
- Ttrain: training latent scores when fit = TRUE. With fit = FALSE, classification routes retain only the compact state needed for prediction or LDA fitting and do not return the full training-score matrix. Compiled cross-validation continues to return its documented score outputs.
- C_latent, W_latent: low-rank latent prediction factors used by PLS-SVD-style compact prediction when a full coefficient array is avoided.
- B: regression coefficient matrix or coefficient array, when stored. For vector-valued ncomp, a three-dimensional array may contain the coefficient path for all requested component counts.
- mX, vX: training predictor centering and scaling values. vX is one when no scaling is applied.
- mY: response centering values for regression or dummy-coded PLS-DA.
- lev: factor levels used for classification.
- Yfit: fitted training responses or fitted class labels, returned when fit = TRUE.
- R2Y: training-set coefficient of determination path when fit = TRUE; otherwise NA placeholders may be returned for compatibility. Elements are named by component count, for example "ncomp=2". For PLS-DA this is a dummy-response quantity, not classification accuracy.
- requested_ncomp: component positions requested by the caller. For rank-limited PLS-LDA fits, every position is retained in fitted and predicted outputs.
- effective_ncomp: number of estimable response-associated directions used for each requested component prefix. Rank-limited PLS-LDA paths repeat the last estimable prediction and discriminant scores. If a regression response is constant, this is zero; fitted and new-data predictions then equal the training-response mean, coefficients are zero, and R2Y is NA.
- Ypred: predictions for Xtest, returned only when Xtest is supplied to pls(). For classification this contains predicted factor labels; for regression it contains numeric predictions.
- Ypred_index: integer class indices for classification predictions, when available.
- Ttest: test-set latent scores, returned when proj = TRUE.
- Q2Y: independent-test Q2 whose denominator centers each response on its corresponding training-response mean before aggregating sums of squares. For factor Ytest, this is dummy-response PLS-DA Q2 relative to training class proportions, not classification accuracy. It is returned when response scores are available. Elements are named by component count.
- accuracy: decoded-label accuracy for factor Ytest, returned when classification predictions are available. Elements are named by component count.
- metrics: complete evaluate() outputs. definitions records the exact R2Y and Q2Y denominator conventions. fitted evaluates Yfit against Ytrain; test evaluates Ypred against Ytest. For multivariate regression, response-wise metrics are included only when bycol = TRUE. metrics$permutation stores permutation metrics and p-values when perm.test = TRUE.

- pval: corrected Monte Carlo permutation-test p-values by component, returned when perm.test = TRUE.
- permutation: long-format permutation table, returned when perm.test = TRUE, with observed and permuted R2/Q2 values and the permutation correlation used by plot.permutation().
- permutation_unit, permutation_group_sizes_preserved, permutation_class_frequencies_preserved, permutation_folds, permutation_solver_seed, permutation_requested, permutation_completed, permutation_failed, and permutation_errors: the permutation contract and null-fit audit.
- variance, variance_explained, cumulative_variance_explained, variance_total, variance_basis: predictor-space variance summaries returned when return_variance = TRUE.
- x_variance, x_variance_explained, x_cumulative_variance_explained, x_variance_total: aliases of the predictor-space variance summaries.
- inner_model: fitted inner PLS model used by OPLS.
- W_orth, P_orth, north, opls_engine, xprod_mode, gpu_resident: OPLS-specific orthogonal-component and backend metadata.
- kernel, kernel_engine, kernel_linear_direct: kernelPLS-specific kernel settings and execution metadata.
- diagnostics: numerical solver diagnostics. For rSVD this records the structural-check status, finiteness, requested and effective component counts and randomized controls. A route that invokes the case-audited CPU decomposition also records its residual audit, strengthened retries, and any deterministic recovery; other routes state explicitly that a case audit is unavailable. Panel evidence is reported separately and is not interpreted as general-use certification. SIMPLS-family fits also record whether the active approximate route uses a component-wise oversampled sketch or an eligible CPU/CUDA/Metal candidate block, together with the active execution optimizations.

Function settings and backend bookkeeping, such as the component grid and resolved classifier backend, are retained internally for prediction and plotting but are not shown as public output fields.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt", "qsec")])
y <- mtcars$mpg
fit <- pls(X, y,
    ncomp = 2, method = "simpls", backend = "cpu",
    return_variance = FALSE
)
head(predict(fit, X)$Ypred)
```

---

pls.double.cv *Nested cross-validation for PLS*

Description

Performs nested grouped cross-validation with an outer loop for unbiased performance estimation and an inner loop for component and hyperparameter selection. Constraint groups are respected in both loops so related samples remain in the same fold.

Usage

```
pls.double.cv(
  Xdata,
  Ydata,
  ncomp = 2,
  constrain = seq_len(nrow(Xdata)),
  scaling = c("centering", "autoscaling", "none"),
  method = c("simpls", "plssvd", "opls", "kernelpls"),
  backend = NULL,
  n.cores = NULL,
  seed = 1L,
  perm.test = FALSE,
  times = 100,
  runn = 1,
  kfold_inner = 10,
  kfold_outer = 10,
  north = 1L,
  kernel = c("linear", "rbf", "poly"),
  gamma = NULL,
  degree = 3L,
  coef0 = 1,
  classifier = c("lda", "argmax"),
  bycol = FALSE,
  selection = "auto",
  return_splits = FALSE,
  ...
)
```

Arguments

`Xdata`	Predictor matrix.
`Ydata`	Response. Use a numeric vector/matrix for regression or factor/character class labels for classification.
`ncomp`	Positive integer component count or vector of counts. Repeated values are removed. When a direct PLS-LDA fit reaches its numerical rank, all requested positions are retained and positions beyond that rank repeat the last estimable prediction and discriminant scores. The fitted object reports both requested and effective component counts.
`constrain`	Grouping vector for grouped cross-validation. It must have one value per sample. Samples with the same value are assigned to the same fold, so all rows from the same patient, subject, batch, or technical replicate stay together in training or test data. The default `seq_len(nrow(Xdata))` treats every sample as an independent group.
`scaling`	One of `centering`, `autoscaling`, or `none`.
`method`	One or more of `simpls`, `plssvd`, `opls`, or `kernelpls`. Multiple values are tuned in the inner loop. `simpls` denotes the fastPLS SIMPLS-family estimator;

an eligible bounded-block route is approximate rather than classical de Jong SIMPLS.

backend — Implementation backend: cpu, cuda, or metal. Multiple values are tuned in the inner loop. Metal requires float32 input and Apple Metal. R validates the request and constructs a reproducible grouped-fold plan. CPU and Metal use the compiled nested-CV coordinator for fold loops, fitting, prediction, and metric accumulation when selection uses accuracy, balanced accuracy or macro recall, Q2Y, or regression RMSD. Other selection metrics use R-level outer coordination around compiled single-CV and fitting kernels. CUDA uses the R coordinator around CUDA-native single-CV and outer-fit kernels; supported CUDA fits do not substitute a CPU estimator. Metal uses the same fixed operation split as pls(). When omitted, options(backend = ...) defines the session default. Every requested backend must be available; unavailable accelerator entries raise an error instead of using CPU.

n.cores — Number of CPU cores requested for supported BLAS/OpenMP host operations. An explicit value takes precedence over options(n.cores = ...). CUDA and Metal device parallelism is managed by their native runtimes; n.cores applies only to their host-side stages.

seed — Random seed used for outer/inner fold assignment and randomized SVD steps.

perm.test — Run a nested-CV permutation test. Independent rows are permuted individually. With repeated constrain values, complete blocks are exchanged only among groups with the same number of rows, preserving group sizes, within-group response structure, and class frequencies. Outer/inner folds and randomized-solver seeds are fixed across observed and null fits. The statistic is the median outer-CV value of the metric used for inner model selection.

times — Number of requested permutations. The corrected Monte Carlo p-value is (b + 1) / (B + 1), using the upper tail for metrics where larger is better and the lower tail for losses such as RMSD. B counts successful null fits only; failed fits are reported. A grouped test requires at least two constraint groups of equal size.

runn — Number of repeated runs.

kfold_inner — Inner-fold count, or "loocv" to leave out one constraint group at a time inside each outer training set.

kfold_outer — Outer-fold count, or "loocv" to leave out one constraint group at a time in the outer loop. In both loops, samples sharing the same constraint value are never split across training and test. A numeric fold count at least as large as the available number of groups has the same leave-one-group-out interpretation.

north — Number of orthogonal components removed by OPLS. The predictive count ncomp must fit within the predictor rank remaining after filtering. Direct PLS-LDA fits retain larger requested positions by repeating the last estimable result; other OPLS routes reject an unavailable predictive direction. Orthogonal components are not counted as predictive components.

kernel — Kernel type for kernel PLS: linear, rbf, or poly.

gamma — Kernel scale. Defaults internally to 1 / ncol(Xdata). For method = "kernelpls", multiple values are tuned in the inner loop.

degree — Polynomial kernel degree.

`coef0` Polynomial kernel offset.

`classifier` Classification decision rule. The default `lda` fits a regularized linear discriminant analysis classifier on the PLS latent scores. `argmax` selects the class with the largest PLS-DA response score.

`bycol` For matrix-valued regression responses, calculate response-wise metrics in the returned `metrics` list. The default `FALSE` returns only aggregate metrics.

`selection` Metric used by inner CV and by the permutation test. `"auto"` uses accuracy for classification and RMSD for regression. Classification also supports `"balanced_accuracy"`, `"AUROC"`, `"lift_accuracy"`, `"macro_precision"`, `"macro_recall"`, `"macro_f1"`, `"kappa"`, `"R2Y"`, and `"Q2Y"`. Regression also supports `"R2Y"`, `"Q2Y"`, `"RMSD"`, `"MAE"`, `"MAPE_percent"`, `"RPD"`, `"Pearson_r"`, and `"Spearman_r"`. R2Y uses fitted responses for inner selection and is averaged across the selected outer training fits for the reported endpoint; as a training criterion, it can favor more complex models. Q2Y and the other predictive criteria use held-out predictions. Q2Y uses fold-training response means; the remaining metrics use the aggregate definitions in `evaluate()`. Binary AUROC selection pools the continuous inner-fold class scores before calculating the criterion; the second factor level is treated as positive. RMSD, MAE, and MAPE_percent are minimized; all other criteria are maximized. R2Y and Q2Y for classification operate on dummy-coded responses. Incompatible task/metric combinations raise an error before fitting. The former names `"r2"` and `"q2"` are rejected as ambiguous.

`return_splits` Return a character matrix named `split_index` when `TRUE`. Rows use the original sample positions. Outer columns contain `"training"` or `"test"`; inner columns additionally use `"outer_test"` for samples excluded by the corresponding outer fold. Column names encode the run and outer/inner fold. The default `FALSE` avoids allocating this additional matrix.

`...` Optional SVD tuning controls forwarded to the selected backend. Use the same compact names documented in `fastsvd()`, such as `oversample` and `power`. Vector values are tuned in the inner loop.

Details

For LDA classification, each inner and outer training fold uses only the classes represented in that fold and maps predictions back to the original factor levels. A class absent from a fold's training data cannot be predicted in that fold; its held-out observations remain in the reported metrics. With CUDA, nested outer/inner orchestration remains in R, while each supported single-CV and outer fit uses its native CUDA route. Unsupported accelerator requests fail explicitly and never fall back to CPU. In either task, a fold that estimates fewer latent components than requested uses its available component prefix, and higher requested prefixes repeat its last estimable prediction and metric. A zero-component regression fold predicts its training-response mean. A zero-component LDA fold uses empirical training-class priors and finite log-prior scores; a single-class training fold predicts its represented class. Inner tuning paths retain every requested component and record fold-level effective counts. Ties are resolved in favor of the earliest, and therefore lowest, requested component count.

Value

A list with the following elements. `metrics$cross_validated` contains one complete `evaluate()` result per repeated outer-CV run, and `metrics$aggregate` evaluates the final vote-aggregated or averaged prediction. For one run, its fold-aware Q2 is also attached to `aggregate`; after several runs, no unique fold-training reference exists for the averaged prediction, so aggregate Q2 is `NA`. `metrics$definitions` records the exact R2Y and Q2Y denominator conventions.

- `results`: list with one element per repeated run. Each run stores `Ypred`/`pred`, the outer `fold` assignment, `best_ncomp` selected in each outer fold, fold-level `best_parameters`, compact inner-CV metric summaries in `inner`, selected outer-fold `effective_ncomp`, run-level `metric_name` and `metric_value`, and the default `backend` and `method`. Each inner summary contains its fold-by-prefix `effective_ncomp` matrix.
- `Ypred`: final cross-validated predictions. For classification, repeated runs are combined by voting; for regression, numeric predictions are averaged across runs.
- `Q2Y`: one outer cross-validated Q2 value per repeated run. Numeric responses use each outer fold's training-response mean. For factor responses this is the mean outer-fold dummy-response PLS-DA Q2 using outer-training class proportions and is not classification accuracy.
- `R2Y`: one training-fit R2 value per repeated run, averaged across the selected outer-fold models.
- `RMSD`: one held-out RMSD value per repeated run for numeric responses; `NA` for classification.
- `metric_name`: metric used for each repeated run. It is held-out unless `selection = "R2Y"` explicitly requests the fitted-response criterion.
- `bcomp`: most frequently selected component count across outer folds and repeated runs.
- `backend`, `method`: default backend and PLS method supplied to the call. If vector-valued methods or backends are tuned, selected fold-level values are stored in `results[[run]]$best_parameters`.
- `selection_metric`: criterion used by the inner CV loop.
- `degenerate_inner_folds`, `constant_classifier_fallback`, `minimum_positive_training_count`, and `minimum_negative_training_count`: audit information for single-class inner-training partitions.
- `component_selection_informative` and `component_selection_informative_by_outer_fold`: whether discrimination could inform component selection globally and in each outer fold.
- `outer_discrimination_estimable` and `non_estimable_outer_folds`: flags for outer-training partitions containing both classes. A single-class outer-training partition returns its constant class rather than terminating the run.
- `acc_tot`: classification-only text summary of correctly classified samples and percentage accuracy.
- `conf`: classification-only confusion matrix printed as counts and column percentages.
- `vote_counts`: classification-only vote-count matrix with one row per sample and one column per class.
- `accuracy`: classification-only decoded-label accuracy, one value per repeated run.
- `balanced_accuracy`: classification-only unweighted mean of class recalls, one value per repeated run.
- `medianR2Y`, `CI95R2Y`, `medianQ2Y`, `CI95Q2Y`, `medianRMSD`, `CI95RMSD`: repeated-run summaries returned only when `runn > 1`.

- permutation_metric, permutation_observed, and permutation_sampled: metric name, observed median, and permuted medians returned when perm.test = TRUE. Q2Ysampled is retained only when Q2 is the selected permutation metric.
- p.value: permutation-test p-value returned when perm.test = TRUE.
- permutation_unit, permutation_group_sizes_preserved, permutation_class_frequencies_preserved, permutation_folds, permutation_solver_seed, permutation_requested, permutation_completed, permutation_failed, and permutation_errors: the exchangeability contract and null-fit audit.
- split_index: sample-by-split membership matrix for every outer and inner fold, returned only when return_splits = TRUE.

Examples

```
idx <- c(seq_len(10), 51:60, 101:110)
X <- as.matrix(iris[idx, seq_len(4)])
y <- factor(iris[idx, 5])
dcv <- pls.double.cv(X, y,
    ncomp = seq_len(2), runn = 1, kfold_inner = 2,
    kfold_outer = 2, method = "simpls", backend = "cpu", seed = 1
)
names(dcv)
```

---

pls.single.cv *Single cross-validation for PLS component optimization*

Description

Performs grouped k-fold or leave-one-out cross-validation over candidate component counts and, when vector-valued predictive arguments are supplied, over a compact hyperparameter grid. Selection uses a task-appropriate metric returned by evaluate() or the fitted-response R2Y path.

Usage

```
pls.single.cv(
  Xdata,
  Ydata,
  ncomp = 2,
  constrain = NULL,
  scaling = c("centering", "autoscaling", "none"),
  method = c("simpls", "plssvd", "opls", "kernelpls"),
  backend = NULL,
  n.cores = NULL,
  seed = 1L,
  kfold = 10,
  north = 1L,
  kernel = c("linear", "rbf", "poly"),
  gamma = NULL,
```

```
  degree = 3L,
  coef0 = 1,
  classifier = c("lda", "argmax"),
  fit = TRUE,
  bycol = FALSE,
  selection = "auto",
  return_splits = FALSE,
  ...
)
```

Arguments

Xdata	Predictor matrix.
Ydata	Response. Use a numeric vector/matrix for regression or factor/character class labels for classification.
ncomp	Positive integer component count or vector of counts. Repeated values are removed. When a direct PLS-LDA fit reaches its numerical rank, all requested positions are retained and positions beyond that rank repeat the last estimable prediction and discriminant scores. The fitted object reports both requested and effective component counts.
constrain	Optional grouping vector for grouped cross-validation. It must have one value per sample. Samples with the same value are assigned to the same fold, so all rows from the same patient, subject, batch, or technical replicate stay together in training or test data. When NULL, each sample is treated as its own group.
scaling	One of centering, autoscaling, or none.
method	One or more of simpls, plssvd, opls, or kernelpls. Multiple values are treated as a tuning grid. simpls denotes the fastPLS SIMPLS-family estimator; an eligible bounded-block route is approximate rather than classical de Jong SIMPLS.
backend	Implementation backend: cpu, cuda, or metal. Multiple values are treated as a tuning grid. Metal requires float32 input and uses fixed CPU/Metal operation splitting with CPU fold orchestration and prediction. When omitted, options(backend = ...) defines the session default. Every requested backend must be available; unavailable accelerator entries raise an error instead of using CPU.
n.cores	Number of CPU cores requested for supported BLAS/OpenMP host operations. An explicit value takes precedence over options(n.cores = ...). CUDA and Metal device parallelism is managed by their native runtimes; n.cores applies only to their host-side stages.
seed	Random seed used for fold assignment and randomized SVD steps.
kfold	Number of folds, or "loocv" for leave-one-out cross-validation. When constrain is supplied, LOOCV means leave-one-constraint-group-out: samples sharing the same constraint value are always held out together and are never split across training and test. A numeric value at least as large as the number of groups has the same leave-one-group-out interpretation.

`north` Number of orthogonal components removed by OPLS. The predictive count `ncomp` must fit within the predictor rank remaining after filtering. Direct PLS-LDA fits retain larger requested positions by repeating the last estimable result; other OPLS routes reject an unavailable predictive direction. Orthogonal components are not counted as predictive components.

`kernel` Kernel type for kernel PLS: `linear`, `rbf`, or `poly`.

`gamma` Kernel scale. Defaults internally to `1 / ncol(Xdata)`. For `method = "kernelpls"`, multiple values are treated as a tuning grid.

`degree` Polynomial kernel degree.

`coef0` Polynomial kernel offset.

`classifier` Classification rule for factor responses. The default `"lda"` fits linear discriminant analysis in the latent score space; `"argmax"` selects the largest PLS-DA response score. Multiple values are treated as a tuning grid.

`fit` Fit one additional model on the full dataset and return its fitted values (`Yfit`) and training `R2Y` path. The default is `TRUE` for backward compatibility. Set to `FALSE` to skip this extra full-data fit; held-out cross-validated `Q2Y` and `RMSD` are still calculated. Selecting `R2Y` overrides `fit = FALSE` because fitted responses are required for that criterion.

`bycol` For matrix-valued regression responses, calculate response-wise metrics in the returned `metrics` list. The default `FALSE` returns only aggregate metrics.

`selection` Metric used to select settings. `"auto"` uses accuracy for classification and RMSD for regression. Classification also supports `"balanced_accuracy"`, `"AUROC"`, `"lift_accuracy"`, `"macro_precision"`, `"macro_recall"`, `"macro_f1"`, `"kappa"`, `"R2Y"`, and `"Q2Y"`. Regression also supports `"R2Y"`, `"Q2Y"`, `"RMSD"`, `"MAE"`, `"MAPE_percent"`, `"RPD"`, `"Pearson_r"`, and `"Spearman_r"`. R2Y uses a model fitted to the complete supplied dataset and therefore forces `fit = TRUE`; as a training criterion, it is usually less suitable for complexity selection than a held-out criterion. Q2Y uses out-of-fold predictions and each fold's training-response mean. The AUROC is available for binary responses and pools continuous held-out class scores across all folds, using the second factor level as the positive class. The remaining metrics use the aggregate definitions in `evaluate()` on the out-of-fold predictions. RMSD, MAE, and MAPE_percent are minimized; all other criteria are maximized. Classification R2Y and Q2Y operate on dummy-coded responses, not decoded labels. Incompatible task/metric combinations raise an error before fitting. The former names `"r2"` and `"q2"` are rejected as ambiguous.

`return_splits` Return a sample-by-fold character matrix named `split_index` when `TRUE`. Rows use the original sample positions and each column represents one fold, with values `"training"` or `"test"`. The default `FALSE` avoids allocating this additional matrix.

`...` Optional SVD tuning controls forwarded to the selected backend. Use the same compact names documented in `fastsvd()`, such as `oversample` and `power`. Vector values are included in the tuning grid.

Details

For LDA classification, each training fold uses only the classes represented in that fold and maps predictions back to the original factor levels. A class absent from a fold's training data cannot be predicted in that fold; its held-out observations remain in the reported metrics. When otherwise identical `"argmax"` and `"lda"` configurations are tuned together, the PLS fit and fold projection are calculated once; each classifier still receives its own predictions, metric path, and selected component count. For sufficiently tall SIMPLS classification problems, full-data predictor and class moments are calculated once and each fold's training moments are obtained by subtracting its held-out contribution. The fold-specific centering, scaling, PLS fit, LDA fit, and predictions remain independent. In either task, a fold that estimates fewer latent components than requested uses its available component prefix, and higher requested prefixes repeat the last estimable prediction and metric. In regression, a zero-component fold predicts its training-response mean. In LDA classification, a zero-component fold uses the empirical training-class priors and finite log-prior discriminant scores; a single-class training fold predicts its represented class. The `effective_ncomp` matrix records the component count used for every fold and requested prefix, while `status` identifies regular, zero-direction, and single-class folds. The returned tuning path always retains every requested component count.

Value

A list describing the cross-validation run and selected model. `metrics$cross_validated` contains complete `evaluate()` results for each requested component count and `metrics$fitted` contains the corresponding full-data fit results when `fit = TRUE`. `metrics$definitions` records the exact R2Y and Q2Y denominator conventions. Other fields are:

- `best_ncomp`: number of components selected by the chosen metric.
- `best_index`: position of `best_ncomp` in the tested component grid.
- `selection_metric`: metric used for optimization. With `"auto"`, classification uses accuracy and regression uses the default prediction error rule.
- `best_metric_name` and `best_metric_value`: name and value of the metric at the selected component count.
- `Q2Y`: held-out cross-validated Q2; every held-out fold is centered on its corresponding fold-training response mean. For factor responses, this is dummy-response PLS-DA Q2 using fold-training class proportions and is not classification accuracy. Values are named by component count.
- `accuracy`: held-out decoded-label accuracy for factor responses, named by component count.
- `balanced_accuracy`: held-out mean class recall for factor responses, named by component count.
- `RMSD`: held-out root mean squared deviation for regression. It is `NA` for classification. Values are named by component count.
- `Yfit`: fitted values from the full-data model when `fit = TRUE`.
- `R2Y`: training-set explained-variance path from a model fitted on the full dataset when `fit = TRUE`; otherwise `NA`. For factor responses, this is calculated on the dummy-coded PLS-DA response scores, not on the decoded class labels.
- `fold`: fold assignment used for each sample.

- `pred`: decoded cross-validated predictions when predictions are stored.
- `Ypred`: raw prediction array when score predictions are stored.
- `lda_scores`: held-out LDA discriminant-score array when LDA scores are stored. For LDA, dummy-response PLS scores are reduced into Q2Y while each fold is processed rather than retained as a second full score array.
- `effective_ncomp`: integer matrix with one row per fold and one column per requested component count. Each entry is the estimable prefix used in that fold. Zero identifies the class-prior fallback.
- `status`: fold status vector. Status 1 is a regular fit, 4 is a single-class training-fold fallback, and 5 is a zero-direction class-prior fallback.
- `degenerate_inner_folds` and `constant_classifier_fallback`: identifiers of single-class training folds and whether the constant-class prediction rule was used.
- `minimum_positive_training_count` and `minimum_negative_training_count`: smallest binary-class counts among fold-training partitions. The second factor level is positive.
- `component_selection_informative`: whether at least one fold could estimate discrimination and the selected metric path contained a finite value.
- `metrics`: complete `evaluate()` outputs. `cross_validated` contains one result per requested component count from held-out predictions and `fitted` contains full-data fit results when `fit = TRUE`. For multivariate regression, response-wise metrics are included only when `bycol = TRUE`.
- `selection_metrics`: compact per-component metric table used for component selection by the CV backend.
- `best_parameters`: compact list containing only `ncomp` plus the arguments that were actually optimized, for example `classifier` when `classifier = c("lda", "argmax")`.
- `tuning_config`: relevant selected configuration used for the run. Irrelevant classifier- or method-specific defaults are omitted; for example, controls belonging to an unselected classifier are omitted.
- `tuning_summary` and `tuning_metrics`: tables for all tested configurations when more than one predictive configuration is supplied.
- `split_index`: sample-by-fold training/test membership matrix, returned only when `return_splits = TRUE`.

Examples

```
idx <- c(seq_len(12), 51:62, 101:112)
X <- as.matrix(iris[idx, seq_len(4)])
y <- factor(iris[idx, 5])
opt <- pls.single.cv(X, y,
    ncomp = seq_len(2), kfold = 3, method = "simpls",
    backend = "cpu", seed = 1
)
opt$best_ncomp
opt_kernel <- pls.single.cv(X, y,
    ncomp = seq_len(2), kfold = 3,
    method = "kernelpls", backend = "cpu",
    kernel = c("linear", "rbf"),
```

```
    gamma = c(0.1, 1), seed = 1
)
opt_kernel$best_parameters
```

---

predict.fastPLS *Predict from fitted fastPLS models*

Description

Generates predictions for new samples from fitted PLS-SVD, SIMPLS-family, OPLS, or kernel PLS models. Stored centering, scaling, latent projections, and model-specific filtering are applied before producing numeric response predictions or classification labels.

Usage

```
## S3 method for class 'fastPLS'
predict(
  object,
  newdata,
  Ytest = NULL,
  proj = FALSE,
  backend = NULL,
  n.cores = NULL,
  top = NULL,
  raw_scores = FALSE,
  ...
)
```

Arguments

object	A fitted fastPLS, fastPLSKernel, or fastPLSOpls object.
newdata	Numeric predictor matrix.
Ytest	Optional observed response. When supplied, the predictions are passed to evaluate() and its complete result is returned in metrics. Regression Q2Y is referenced to the response mean stored during model training.
proj	Logical; return projected Ttest when TRUE.
backend	Prediction backend: "cpu", "cuda", or "metal". Operation-split Metal models predict on CPU because their retained matrices are host-accessible; Metal is used for fitting sample-matrix products. When omitted, an explicit session backend setting is used, followed by the FASTPLS_BACKEND environment variable; otherwise CPU is used. Prediction must use the backend that fitted the model. An unavailable CUDA or Metal selection raises an error; prediction is never silently moved to CPU.
n.cores	Number of CPU cores requested for compiled host operations. An explicit value takes precedence over options(n.cores = ...). This controls supported BLAS/OpenMP host work and does not set CUDA or Metal device parallelism.

top	Number of ranked classes to return for classification. The default NULL returns only the predicted class in Ypred. A positive integer greater than one additionally returns that many ordered classes per sample in Ypred_top; top = 5, for example, returns five classes. This argument is ignored with a warning for regression models.
raw_scores	If TRUE, keep raw classification score cubes as Yscore when available. This can require substantial memory. With raw_scores = FALSE, ranked classification is evaluated in bounded row blocks for both float64 and float32 inputs, and only the requested ranks are retained.
...	Required by the S3 generic. Additional arguments are not supported and produce an error, which prevents obsolete or misspelled options from being silently ignored.

Value

A list containing Ypred, optional independent-test Q2Y, optional Ttest, optional Ypred_top and Ypred_top_score ranked-class outputs, and optional raw classification scores. When Ytest is supplied, metrics contains the complete result returned by evaluate() for every requested component count. For a rank-limited PLS-LDA fit, the prediction path retains every requested position and repeats the last estimable class prediction and discriminant scores.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt", "qsec")])
y <- mtcars$mpg
fit <- pls(X, y,
    ncomp = 2, method = "simpls", backend = "cpu",
    return_variance = FALSE
)
pred <- predict(fit, X[seq_len(3), , drop = FALSE])
pred$Ypred
```

ViP *Variable importance in projection (VIP)*

Description

Computes VIP trajectories from fitted direct SIMPLS-family components. The standard component-wise decomposition is not used for PLS-SVD, OPLS, or nonlinear kernel PLS because their stored latent weights have different mathematical meanings. Linear-kernel PLS uses the same direct SIMPLS-family path and is supported.

Usage

```
ViP(model)
```

Arguments

model	Fitted fastPLS model.

Value

Numeric matrix (single response) or list of matrices (multi-response).

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt", "qsec")])
y <- mtcars$mpg
fit <- pls(X, y,
    ncomp = 1, method = "simpls", backend = "cpu",
    fit = TRUE, return_variance = FALSE
)
ViP(fit)
```

Index

cuda_info, 2

evaluate, 3
evaluate(), *18*, *20*, *22*

fastcor, 5
fastPLS_blas, 6
fastPLS_blas(), *13*
fastsvd, 7
fastsvd(), *13*, *18*, *22*

gpuinfo::cuda_info(), *3*
gpuinfo::cuda_libraries(), *3*

legend(), *10*

plot(), *10*
plot.fastPLS, 9
plot.permutation, 10
pls, *3*, 11
pls.double.cv, *5*, 15
pls.single.cv, *5*, 20
predict.fastPLS, 25

ViP, 26