Package 'savvyPR'

June 9, 2026

Type Package

Title Savvy Parity Regression Model Estimation with 'savvyPR'

Version 0.1.2

Description Implements the Savvy Parity Regression 'savvyPR' methodology for multivariate linear regression analysis. The package solves an optimization problem that balances the contribution of each predictor variable to ensure estimation stability in the presence of multicollinearity. It supports two distinct parameterization methods, a Budget-based approach that allocates a fixed loss contribution to each predictor, and a Target-based approach (t-tuning) that utilizes a relative elasticity weight for the response variable. The package provides comprehensive tools for model estimation, risk distribution analysis, and parameter tuning via cross-validation (PR1, PR2, and PR3 model types) to optimize predictive accuracy. Methods are based on Asimit, Chen, Ichim and Millossovich (2026) <https://openaccess.city.ac.uk/id/eprint/37017/>.

License GPL (>= 3)

URL https://ziwei-chenchen.github.io/savvyPR/

BugReports https://github.com/ziwei-chenchen/savvyPR/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.6.0)

Imports glmnet, Matrix, stats, nleqslv, ggplot2, gridExtra

Suggests MASS, knitr, rmarkdown, testthat (>= 3.0.0), covr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Ziwei Chen [aut, cre] (ORCID: <https://orcid.org/0009-0009-6376-3850>), Vali Asimit [aut] (ORCID: <https://orcid.org/0000-0002-7706-0066>), Pietro Millossovich [aut] (ORCID: <https://orcid.org/0000-0001-8269-7507>)

Maintainer Ziwei Chen <Ziwei.Chen.3@citystgeorges.ac.uk>

Repository CRAN

Date/Publication 2026-06-09 20:00:02 UTC

Contents

`coef.cv.savvyPR` *Extract Coefficients From a Cross-Validated Parity Regression Model Object*

Description

Extracts the estimated coefficients corresponding to the optimal tuning parameters from a fitted cross-validated parity regression model object. It seamlessly handles models optimized using either the `"budget"` or `"target"` parameterization method.

Usage

```
## S3 method for class 'cv.savvyPR'
coef(object, ...)
```

Arguments

`object`	A fitted model object of class `cv.savvyPR`.
`...`	Additional arguments passed to the `predict.cv.savvyPR` method.

Details

Extract Coefficients From a Cross-Validated Parity Regression Model Object

This function is an S3 method for the generic `coef` function. It extracts the coefficients of the optimal model identified during the cross-validation procedure. The extraction is internally delegated to the `predict.cv.savvyPR` method with `type = "coefficients"`.

Value

A named numeric vector of the estimated coefficients from the optimally tuned model. If the model was fitted with an intercept, it will be included as the first element.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

`cv.savvyPR`, `predict.cv.savvyPR`

Examples

```
# Generate synthetic data
set.seed(123)
x <- matrix(rnorm(100 * 10), 100, 10)
y <- rnorm(100)

# Example 1: Extract coefficients from a Budget-based CV model
cv_model_budget <- cv.savvyPR(x, y, method = "budget", model_type = "PR3")
coef(cv_model_budget)

# Example 2: Extract coefficients from a Target-based CV model
cv_model_target <- cv.savvyPR(x, y, method = "target", model_type = "PR1")
coef(cv_model_target)
```

---

coef.savvyPR *Extract Coefficients From a Parity Regression Model Object*

Description

Extracts the estimated coefficients from a fitted parity regression model object. It seamlessly handles models fitted using either the `"budget"` or `"target"` parameterization method.

Usage

```
## S3 method for class 'savvyPR'
coef(object, ...)
```

Arguments

`object`	A fitted model object of class `savvyPR`.
`...`	Additional arguments passed to the `predict.savvyPR` method.

Details

Extract Coefficients From a Parity Regression Model Object

This function is an S3 method for the generic `coef` function. It extracts the estimated coefficients of the fitted model. The extraction is internally delegated to the `predict.savvyPR` method with `type = "coefficients"`.

Value

A named numeric vector of the estimated coefficients from the fitted model. If the model was fitted with an intercept, it will be included as the first element.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

`savvyPR`, `predict.savvyPR`

Examples

```
# Generate synthetic data
set.seed(123)
x <- matrix(rnorm(100 * 10), 100, 10)
y <- rnorm(100)

# Example 1: Extract coefficients from a Budget-based model
model_budget <- savvyPR(x, y, method = "budget", val = 0.05)
coef(model_budget)

# Example 2: Extract coefficients from a Target-based model
model_target <- savvyPR(x, y, method = "target", val = 1)
coef(model_target)
```

---

cv.savvyPR *Cross-Validation for Parity Regression Model Estimation*

Description

Performs k-fold cross-validation for Parity Regression (PR) models to select optimal tuning parameters. The underlying PR methodology distributes the total prediction error evenly across all parameters, ensuring stability in the presence of high multicollinearity and substantial noise (such as time series data with structural changes and evolving trends). This function supports both Budget-based and Target-based parameterizations and evaluates models across a variety of loss metrics.

Usage

```
cv.savvyPR(
  x,
  y,
  method = c("budget", "target"),
  vals = NULL,
  nval = 100,
  lambda_vals = NULL,
  nlambda = 100,
  folds = 10,
  model_type = c("PR3", "PR1", "PR2"),
  measure_type = c("mse", "mae", "rmse", "mape"),
  foldid = FALSE,
  use_feature_selection = FALSE,
  standardize = FALSE,
  intercept = TRUE,
  exclude = NULL
)
```

Arguments

x	A matrix of predictors with rows as observations and columns as variables. Must not contain NA values, and should not include an intercept column of ones.
y	A numeric vector of the response variable, should have the same number of observations as x. Must not contain NA values.
method	Character string specifying the parameterization method to use: "budget" (default) or "target".
vals	Optional; a numeric vector of values for tuning the PR model (acts as c for budget, or t for target). If NULL, a default sequence is generated based on the selected method. Must contain at least two values.
nval	Numeric value specifying the number of tuning values to try in the optimization process if vals=NULL. Defaults to 100.
lambda_vals	Optional; a numeric vector of lambda values used for regularization in the "PR2" and "PR3" model types. If NULL and model_type is "PR2" or "PR3", a default sequence is used. Must contain at least two values.
nlambda	Numeric value specifying the number of lambda_val values to try in the optimization process if lambda_vals=NULL.
folds	The number of folds to be used in the cross-validation, default is 10. Must be an integer >= 3.
model_type	Character string specifying the type of model to fit. Defaults to "PR3". Can be one of "PR3", "PR1", or "PR2". See details for further clarification.
measure_type	Character vector specifying the measure to use for model evaluation. Defaults to "mse". Supported types include "mse", "mae", "rmse", and "mape".
foldid	Logical indicating whether to return fold assignments. Defaults to FALSE.

`use_feature_selection`
: Logical indicating whether to perform feature selection during the model fitting process. Defaults to `FALSE`.

`standardize`
: Logical indicating whether to standardize predictor variables. Defaults to `TRUE`.

`intercept`
: Logical indicating whether to include an intercept in the model. Defaults to `TRUE`.

`exclude`
: Optional; indicate if any variables should be excluded in the model fitting process.

Details

Cross-Validation for Parity Regression Model Estimation

This function facilitates cross-validation for parity regression models across a range of tuning values (`val`) and regularization values (λ), depending on the model type specified. Each model type handles the parameters differently:

PR1 Performs cross-validation only over the `val` sequence while fixing $\lambda = 0$ (Standard OLS baseline). **Note:** In high-dimensional settings ($p \geq n$), an OLS baseline is invalid. In this scenario, PR1 automatically enforces a minimal Ridge penalty ($\lambda = 1e-4$) to maintain mathematical stability while still focusing purely on the parity mechanism.

PR2 Uses a fixed λ value determined by performing a ridge regression (`lambda` optimization) using `cv.glmnet` on the dataset. It then performs cross-validation over the `val` sequence while using this optimized λ value. This approach is useful when one wishes to maintain a stable amount of standard shrinkage while exploring the impact of varying levels of the proportional contribution constraint. Ensures $\lambda > 0$ if $p \geq n$.

PR3 First, determines an optimal `val` using the same method as `PR1`. Then, keeping this `val` fixed, it conducts a cross-validation over all possible λ values. This dual-stage optimization can be particularly effective when the initial parity regularization needs further refinement via λ adjustment. Automatically filters out $\lambda = 0$ in high-dimensional datasets.

The function supports several types of loss metrics for assessing model performance:

mse Mean Squared Error: Measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

mae Mean Absolute Error: Measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

rmse Root Mean Squared Error: It is the square root of the mean of the squared errors. `RMSE` is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

mape Mean Absolute Percentage Error: Measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error, as shown above. Because it is based on relative errors, it is less sensitive to large deviations in small true values.

The choice of measure impacts how the model's performance is assessed during cross-validation. Users should select the measure that best reflects the requirements of their specific analytical context.

Value

A list of class "cv.savvyPR" containing the following components based on the specified model_type:

call	The matched call used to invoke the function.
coefficients	The optimal coefficients results of the final fitted model.
mean_error_cv	A vector of computed error values across all tested parameters.
model_type	The type of PR model used: PR1, PR2, or PR3.
measure_type	The loss measure used for evaluation, with a descriptive name.
method	The parameterization method used: "budget" or "target".
PR_fit	The final fitted model object from the savvyPR function.
coefficients_cv	A matrix of average coefficients across all cross-validation folds for each tuning parameter.
vals	The tuning values (acting as c or t) used in the cross-validation process.
lambda_vals	The lambda values used in the cross-validation process, applicable to PR2 and PR3.
optimal_val	The optimal tuning value found from cross-validation, applicable to PR1 and PR2.
fixed_val	The fixed tuning value used in PR3, derived from an initial PR1-style optimization.
optimal_lambda_val	The optimal lambda value found in PR3.
fixed_lambda_val	The fixed lambda value used in PR2, derived from cv.glmnet.
optimal_index	A list detailing the indices of the optimal parameters within the cross-validation matrix.
fold_assignments	(Optional) The fold assignments used during the cross-validation, provided if foldid=TRUE.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

References

Asimit, V., Chen, Z., Ichim, B., & Millossovich, P. (2026). *Parity Regression Estimation*. Retrieved from https://openaccess.city.ac.uk/id/eprint/37017/

The optimization technique employed follows the algorithm described by: F. Spinu (2013). An Algorithm for Computing Risk Parity Weights. SSRN Preprint. doi:10.2139/ssrn.2297383

See Also

savvyPR, glmnet, cv.glmnet, calcLoss, getMeasureName, optimizeRiskParityBudget, optimizeRiskParityTarget

Examples

```
# Generate synthetic data
set.seed(123)
n <- 10 # Number of observations
p <- 12  # Number of variables
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5)

# Example 1: PR1 with "budget" method (focusing on c values with MSE)
result_pr1_budget <- cv.savvyPR(x, y, method = "budget", model_type = "PR1")
print(result_pr1_budget)

# Example 2: PR1 with "target" method
result_pr1_target <- cv.savvyPR(x, y, method = "target", model_type = "PR1")
print(result_pr1_target)

# Example 3: PR3 (default model_type) exploring budget parameter
result_pr3 <- cv.savvyPR(x, y, method = "budget", folds = 5)
print(result_pr3)
```

`plot.cv.savvyPR` *Plot for a Cross-Validated Parity Regression Model*

Description

Generates various visualizations for a fitted cross-validated parity regression model object. It supports plotting estimated coefficients, risk contributions, coefficient paths, and cross-validation error curves based on the specified `plot_type`.

Usage

```
## S3 method for class 'cv.savvyPR'
plot(
  x,
 plot_type = c("estimated_coefficients", "risk_contributions", "cv_coefficients",
    "cv_errors"),
  label = TRUE,
  xvar = c("norm", "lambda", "dev", "val"),
  max_vars_per_plot = 10,
  ...
)
```

Arguments

`x`	A fitted model object of class `"cv.savvyPR"` returned by `cv.savvyPR`.

`plot_type` Character string specifying the type of plot to generate. Can be `"estimated_coefficients"`, `"risk_contributions"`, `"cv_coefficients"`, or `"cv_errors"`. Defaults to `"estimated_coefficients"`.

`label` Logical; if `TRUE`, labels are added based on the plot type:

cv_coefficients Variable names are added to the coefficient paths.

risk_contributions Numeric labels are added above the bars.

estimated_coefficients Numeric values are added to the coefficient plot.

cv_errors Numeric labels are added to the optimal error point.

Default is `TRUE`.

`xvar` Character string specifying the x-axis variable for plotting coefficient paths. Options are `"norm"`, `"lambda"`, `"dev"`, and `"val"`. This argument is only used when `plot_type = "cv_coefficients"`.

`max_vars_per_plot` Integer specifying the maximum number of variables to plot per panel. Cannot exceed `10`. Default is `10`. This argument is only used when `plot_type = "cv_coefficients"`.

`...` Additional arguments passed to the underlying `ggplot` function.

Details

Plot for a Cross-Validated Parity Regression Model

This function offers four types of plots, depending on the value of `plot_type`:

Estimated Coefficients Generates a line plot with points for the estimated coefficients of the optimally tuned cross-validated regression model. If an intercept term is included, it will be labeled as `beta_0`; otherwise, coefficients are labeled sequentially based on the covariates. If `label = TRUE`, numeric values are displayed on the plot. This plot helps visualize the contribution of each predictor variable to the model.

Risk Contributions Generates two bar plots, one for the optimization variables (weights or target values) and one for risk contributions, from the risk parity model. If `label = TRUE`, numeric labels are added above the bars for clarity.

Coefficient Paths Generates a plot showing the coefficient paths against the selected `x-axis` variable (`val`, `lambda`, `norm`, or `dev`) depending on the model type:

- **PR1/PR2:** Plots coefficient paths against `log(val)` values.
- **PR3:** Plots coefficient paths against `log(lambda)` values.

Invalid combinations of `model_type` and `xvar` will result in an error:

- **PR1/PR2:** Cannot use `"lambda"` as `xvar`.
- **PR3:** Cannot use `"val"` as `xvar`.

If `max_vars_per_plot` exceeds `10`, it is reset to `10`. The plot provides insight into how coefficients evolve across different regularization parameters.

Cross-Validation Errors Generates a plot that shows the cross-validation error metric against the logarithm of the tuning parameter (`val` or `lambda`), depending on the model type. It adds a vertical dashed line to indicate the optimal parameter value.

- **PR1/PR2:** Plots cross-validation errors against `log(val)` values.
- **PR3:** Plots cross-validation errors against `log(lambda)` values.

Value

A ggplot object representing the requested plot.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

cv.savvyPR

Examples

```
# Example usage for `cv.savvyPR` with Correlated Data:
set.seed(123)
n <- 100
p <- 10
# Create highly correlated predictors to demonstrate parity regression
base_var <- rnorm(n)
x <- matrix(rnorm(n * p, sd = 0.1), n, p) + base_var
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5)

# Fit CV model using Budget method
cv_result1 <- cv.savvyPR(x, y, method = "budget", model_type = "PR1",
                         measure_type = "mse", intercept = FALSE)
plot(cv_result1, plot_type = "estimated_coefficients")
plot(cv_result1, plot_type = "risk_contributions", label = FALSE)
plot(cv_result1, plot_type = "cv_coefficients", xvar = "val", max_vars_per_plot = 10)
plot(cv_result1, plot_type = "cv_errors")

# Fit CV model using Target method
cv_result2 <- cv.savvyPR(x, y, method = "target", model_type = "PR2")
cv_result3 <- cv.savvyPR(x, y, method = "budget", model_type = "PR3")

plot(cv_result2, plot_type = "cv_coefficients", xvar = "val",
     max_vars_per_plot = 5, label = FALSE)
plot(cv_result3, plot_type = "cv_coefficients", xvar = "lambda",
     max_vars_per_plot = 10, label = TRUE)
plot(cv_result2, plot_type = "cv_errors", label = FALSE)
plot(cv_result3, plot_type = "cv_errors", label = TRUE)
```

---

plot.savvyPR *Plot for a Parity Regression Model*

Description

Generates various visualizations for a fitted parity regression model object. It supports plotting estimated coefficients and risk contributions based on the specified `plot_type`.

Usage

```
## S3 method for class 'savvyPR'
plot(
  x,
  plot_type = c("estimated_coefficients", "risk_contributions"),
  label = TRUE,
  ...
)
```

Arguments

`x`	A fitted model object of class `"savvyPR"` returned by `savvyPR`.
`plot_type`	Character string specifying the type of plot to generate. Can be `"estimated_coefficients"` or `"risk_contributions"`. Defaults to `"estimated_coefficients"`.
`label`	Logical; if `TRUE`, labels are added based on the plot type: **estimated_coefficients** Numeric values are added to the coefficient plot. **risk_contributions** Numeric labels are added above the bars. Default is `TRUE`.
`...`	Additional arguments passed to the underlying `ggplot` function.

Details

Plot for a Parity Regression Model

This function offers two types of plots, depending on the value of `plot_type`:

Estimated Coefficients Generates a line plot with points for the estimated coefficients of the regression model. If an intercept term is included in the model, it will be labeled as `beta_0`. Otherwise, the coefficients are labeled sequentially as `beta_1`, `beta_2`, etc., based on the covariates. This plot helps to visualize the contribution of each predictor variable to the model. If `label = TRUE`, numeric values are displayed.

Risk Contributions If the model includes a risk parity component, the function will check if the optimization results (e.g., `orp_fit$weights` for the budget method, or `orp_fit$x` for the target method, along with `orp_fit$relativeRiskContrib`) are available. If available, two bar plots are created:

- **Optimization Variables:** A bar plot that visualizes the optimal variables assigned to each covariate and the response variable (weights for budget, target parameters for target).
- **Risk Contributions:** A bar plot that visualizes the relative risk contributions of each covariate and the response variable.

If `label = TRUE`, numeric labels are added above the bars for clarity. If they are not found, a warning is issued.

Value

A ggplot object representing the requested plot.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

savvyPR

Examples

```
# Example usage for `savvyPR` with Correlated Data:
set.seed(123)
n <- 100
p <- 12
# Create highly correlated predictors to demonstrate parity regression
base_var <- rnorm(n)
x <- matrix(rnorm(n * p, sd = 0.1), n, p) + base_var
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5)

# Fit a Budget-based parity regression model
result_budget <- savvyPR(x, y, method = "budget", val = 0.05, intercept = TRUE)
plot(result_budget, plot_type = "estimated_coefficients", label = FALSE)
plot(result_budget, plot_type = "risk_contributions", label = TRUE)

# Fit a Target-based parity regression model
result_target <- savvyPR(x, y, method = "target", val = 1, intercept = TRUE)
plot(result_target, plot_type = "risk_contributions", label = TRUE)
```

---

predict.cv.savvyPR *Predict for Cross-Validated Parity Regression Models*

Description

Predicts fitted values or extracts estimated coefficients from a fitted cross-validated parity regression model object. It handles models optimized using either the "budget" or "target" parameterization method seamlessly.

Usage

```
## S3 method for class 'cv.savvyPR'
predict(object, newx = NULL, type = c("response", "coefficients"), ...)
```

Arguments

`object`	A fitted model object of class `"cv.savvyPR"` returned by `cv.savvyPR`.
`newx`	Matrix of new data for which predictions are to be made. Must have the same number of columns as the training data. This argument is required if `type = "response"`.
`type`	Type of prediction required. Can be `"response"` (fitted values) or `"coefficients"`. Defaults to `"response"`.
`...`	Additional arguments (currently unused in this function).

Details

Predict for Cross-Validated Parity Regression Models

This function is an S3 method for the generic `predict` function. It utilizes the optimal model identified during the cross-validation procedure. For `type = "response"`, it computes predictions based on the provided `newx` matrix. For `type = "coefficients"`, it extracts the optimal coefficients. The underlying computation is delegated to an internal helper function to ensure consistency across the package.

Value

Depending on the `type` argument, this function returns:

- `"response"`: A numeric vector of predicted values corresponding to the rows of `newx`.
- `"coefficients"`: A named numeric vector of the estimated optimal coefficients. If the model was fitted with an intercept, it will be included as the first element.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

`cv.savvyPR`

Examples

```
# Generate synthetic data
set.seed(123)
x <- matrix(rnorm(100 * 20), 100, 20)
y <- rnorm(100)

# Example 1: Predict using a cross-validated Budget-based model
cv_fit_budget <- cv.savvyPR(x, y, method = "budget", model_type = "PR3")
predict(cv_fit_budget, newx = x[1:5, ], type = "response")

# Example 2: Predict using a cross-validated Target-based model
cv_fit_target <- cv.savvyPR(x, y, method = "target", model_type = "PR1")
predict(cv_fit_target, newx = x[1:5, ], type = "response")
```

```
# Extract optimal coefficients
predict(cv_fit_budget, type = "coefficients")
```

`predict.savvyPR` *Predict for Parity Regression Models*

Description

Predicts fitted values or extracts estimated coefficients from a fitted parity regression model object. It handles models optimized using either the `"budget"` or `"target"` parameterization method seamlessly.

Usage

```
## S3 method for class 'savvyPR'
predict(object, newx = NULL, type = c("response", "coefficients"), ...)
```

Arguments

`object`	A fitted model object of class `"savvyPR"` returned by `savvyPR`.
`newx`	Matrix of new data for which predictions are to be made. Must have the same number of columns as the training data. This argument is required if `type = "response"`.
`type`	Type of prediction required. Can be `"response"` (fitted values) or `"coefficients"`. Defaults to `"response"`.
`...`	Additional arguments (currently unused in this function).

Details

Predict for Parity Regression Models

This function is an S3 method for the generic `predict` function. It utilizes the parameters estimated during the fitting procedure. For `type = "response"`, it computes predictions based on the provided `newx` matrix. For `type = "coefficients"`, it extracts the estimated coefficients. The underlying computation is delegated to an internal helper function to ensure consistency across the package.

Value

Depending on the `type` argument, this function returns:

- `"response"`: A numeric vector of predicted values corresponding to the rows of `newx`.
- `"coefficients"`: A named numeric vector of the estimated coefficients. If the model was fitted with an intercept, it will be included as the first element.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

savvyPR

Examples

```
# Generate synthetic data
set.seed(123)
x <- matrix(rnorm(100 * 20), 100, 20)
y <- rnorm(100)

# Example 1: Predict using a Budget-based model
fit_budget <- savvyPR(x, y, method = "budget", val = 0.05)
predict(fit_budget, newx = x[1:5, ], type = "response")

# Example 2: Predict using a Target-based model
fit_target <- savvyPR(x, y, method = "target", val = 1)
predict(fit_target, newx = x[1:5, ], type = "response")

# Extract coefficients
predict(fit_budget, type = "coefficients")
```

---

print.cv.savvyPR *Print a Cross-Validated Parity Regression Model Object*

Description

Prints a summarized output of a fitted cross-validated parity regression model object. It clearly displays the optimal tuning parameters and the resulting estimated coefficients.

Usage

```
## S3 method for class 'cv.savvyPR'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	A fitted model object of class "cv.savvyPR" returned by cv.savvyPR.
digits	Significant digits to be used in the printout.
...	Additional arguments passed to the generic print function.

Details

Print a Cross-Validated Parity Regression Model Object

This function is an S3 method for the generic `print` function. It formats and prints the matched call that produced the `cv.savvyPR` object, followed by a summary data frame. This summary includes:

- The parameterization method used (`"budget"` or `"target"`).
- The number of non-zero coefficients.
- Whether an intercept was included.
- The optimal tuning value (`val`) and/or `lambda` parameter, depending on the `model_type` (`PR1`, `PR2`, or `PR3`).

Finally, it prints a data frame of the optimally tuned estimated coefficients.

Value

Invisibly returns a data frame summarizing the cross-validation results, including the parameterization method, number of non-zero coefficients, and optimal tuning parameters.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

`cv.savvyPR`

Examples

```
# Generate synthetic data
set.seed(123)
n <- 100
p <- 10
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5)

# Fit and print a cross-validated budget-based parity regression model
cv_fit_budget <- cv.savvyPR(x, y, method = "budget", model_type = "PR3")
print(cv_fit_budget)

# Fit and print a cross-validated target-based parity regression model
cv_fit_target <- cv.savvyPR(x, y, method = "target", model_type = "PR1")
print(cv_fit_target)
```

`print.savvyPR` *Print a Parity Regression Model Object*

Description

Prints a summarized output of a fitted parity regression model object. It clearly displays the model's configuration and the resulting estimated coefficients.

Usage

```
## S3 method for class 'savvyPR'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	A fitted model object of class `"savvyPR"` returned by `savvyPR`.
`digits`	Significant digits to be used in the printout.
`...`	Additional arguments passed to the generic `print` function.

Details

Print a Parity Regression Model Object

This function is an S3 method for the generic `print` function. It formats and prints the matched call that produced the `savvyPR` object, followed by a summary data frame. This summary includes:

- The parameterization method used (`"budget"` or `"target"`).
- The number of non-zero coefficients.
- Whether an intercept was included.
- The penalty value (`lambda`), if any was applied.

Finally, it prints a data frame of the estimated coefficients.

Value

Invisibly returns a data frame summarizing the model, including the parameterization method, number of non-zero coefficients, intercept status, and lambda value.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

`savvyPR`

Examples

```
# Generate synthetic data
set.seed(123)
n <- 100
p <- 10
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5)

# Fit and print a Budget-based model
fit_budget <- savvyPR(x, y, method = "budget", val = 0.05)
print(fit_budget)

# Fit and print a Target-based model
fit_target <- savvyPR(x, y, method = "target", val = 1)
print(fit_target)
```

---

savvyPR *Parity Regression Model Estimation*

Description

Implements the Parity Regression (PR) methodology for multiple linear regression. Instead of minimizing the aggregate prediction error in the dependent variable, PR distributes the total prediction error evenly across all parameters. This approach ensures stability in the presence of high multicollinearity and is particularly suitable for data affected by substantial noise, such as time series data experiencing structural changes and evolving trends.

Usage

```
savvyPR(
  x,
  y,
  method = c("budget", "target"),
  val = NULL,
  lambda_val = NULL,
  use_feature_selection = FALSE,
  standardize = FALSE,
  intercept = TRUE,
  exclude = NULL
)
```

Arguments

x	A matrix of predictors with rows as observations and columns as variables. Must not contain NA values, and should not include an intercept column of ones.

y	A numeric vector of the response variable, should have the same number of observations as x. Must not contain NA values.
method	Character string specifying the parameterization method to use: "budget" (default) or "target".
val	Numeric tuning parameter. If method = "budget", this represents c (a value between 0 and 1/p). If method = "target", this represents t (a target risk parameter > 0).
lambda_val	Optional; a numeric value specifying the regularization strength. If NULL, lambda is determined via cross-validation. **Note:** In high-dimensional settings ($p \geq n$), this value must be strictly greater than 0 to prevent the loss function from reaching zero.
use_feature_selection	Logical; if TRUE, applies Lasso to perform feature selection before model estimation. Defaults to FALSE.
standardize	Logical; if TRUE, scale and center predictor variables before fitting the model. Defaults to TRUE.
intercept	Logical; if TRUE, includes an intercept in the model, otherwise, the model will not estimate an intercept term. Defaults to TRUE.
exclude	Optional; an array of indices specifying columns to be excluded from the analysis.

Details

Parity Regression Model Estimation

The PR methodology assigns an equal risk contribution constraint on each predictor variable within a specific search cone, leading to a robust solution. This solution is defined in the context of a penalized regression, wherein the penalty term is a function of both the regularization parameter (lambda_val) and the proportional contribution parameter (val). The function uses the nleqslv package to solve non-linear systems of equations.

The function supports two parameterization methods:

- **Budget**: Uses val (acting as c) to set a strict budget constraint on the risk contributions.
- **Target**: Uses val (acting as t) to set a risk target for the response variable relative to the predictors.

The function can handle different practical scenarios:

- While the PR theorem is not specifically designed for variable selection, the package makes this available as an optional preprocessing step. If use_feature_selection is TRUE, Lasso regression is performed to select features by zeroing out non-contributive predictors before applying the PR model.
- It fully supports high-dimensional ($p \geq n$) and rank-deficient datasets. In these scenarios, the function applies Ridge regression as a fallback to ordinary least squares to establish the initial parameter signs (the search cone). A strictly positive lambda_val is enforced to ensure computational stability and maintain the validity of the logarithmic barrier constraints.

For the **budget** method, the PR methodology optimizes an objective function that includes a variance term and a penalization term:

$$1/2 * RRSS(x, x_{p+1}; \lambda) - \widetilde{\mu}(c \sum_{k=0}^{p} \log(\delta_k x_k) + (1 - (p+1)c) \log(x_{p+1}))$$

For the **target** method, the methodology optimizes a related objective function defined by the target parameter t:

$$1/2 * RRSS(x, x_{p+1}; \lambda) - \widetilde{\mu}(\sum_{k=0}^{p} \log(\delta_k x_k) + t \log(x_{p+1}))$$

In both formulas, x represents the parameters (with x_{p+1} as an auxiliary parameter set to 1), λ is the regularization parameter, p is the number of predictors, and $\widetilde{\mu}$ is a constant with respect to λ. The resulting model provides estimates of the regression coefficients that are equitable across all predictors in terms of contribution to the model's predictive power.

Value

Returns an S3 object of class `"savvyPR"` containing the following components:

`call`	The matched call to the function.
`coefficients`	A numeric vector of estimated coefficients. If the tuning parameter is zero, coefficients are obtained from Ridge regression or OLS. Otherwise, they are obtained from the parity regression model.
`method`	The optimization method used (`"budget"` or `"target"`).
`fit`	The fitted object returned by `glmnet` when the tuning parameter is zero.
`orp_fit`	The fitted object returned by the risk parity optimizer (`optimizeRiskParityBudget` or `optimizeRiskParityTarget`) when the tuning parameter is non-zero.
`lambda`	The regularization parameter `lambda` used in the model.
`intercept`	A logical value indicating whether an intercept is included in the model.
`model`	A data frame containing the response variable `y` and the covariates `x` used in the model.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

References

Asimit, V., Chen, Z., Ichim, B., & Millossovich, P. (2026). *Parity Regression Estimation*. Retrieved from https://openaccess.city.ac.uk/id/eprint/37017/

The optimization technique employed follows the algorithm described by: F. Spinu (2013). An Algorithm for Computing Risk Parity Weights. SSRN Preprint. doi:10.2139/ssrn.2297383

See Also

`cv.savvyPR`, `glmnet`, `cv.glmnet`

Examples

```
library(glmnet)
library(nleqslv)

# Generate synthetic data
set.seed(123)
n <- 100 # Number of observations
p <- 10  # Number of variables
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5) # Linear combination with noise

# Example 1: Run PR estimation using the "budget" method (acting as c)
result_budget <- savvyPR(x, y, method = "budget", val = 0.05, intercept = TRUE)
print(result_budget$coefficients)

# Example 2: Run PR estimation using the "target" method (acting as t)
result_target <- savvyPR(x, y, method = "target", val = 1, intercept = TRUE)
print(result_target$coefficients)

# Example 3: Run PR estimation with feature selection
result_fs <- savvyPR(x, y, method = "budget", val = 0.05, use_feature_selection = TRUE)
print(result_fs$coefficients)

# Inspect the risk parity portfolio object for more details
if(!is.null(result_fs$orp_fit)) {
  print("Risk parity portfolio details:")
  print(result_fs$orp_fit)
}

# Example 4: Run PR estimation excluding some predictors
result_exclude <- savvyPR(x, y, method = "budget", val = 0.05, exclude = c(1, 2))
print("Coefficients with first two predictors excluded:")
print(result_exclude$coefficients)
```

`summary.cv.savvyPR` *Summary of Cross-Validated Parity Regression Model*

Description

Prints a comprehensive statistical summary of a fitted cross-validated parity regression model object. It displays the model's configuration, statistical summaries of the estimated coefficients, model fit statistics, and cross-validation results.

Usage

```
## S3 method for class 'cv.savvyPR'
summary(object, ...)
```

Arguments

object	A fitted model object of class cv.savvyPR returned by cv.savvyPR.
...	Additional arguments passed to the generic summary function (currently unused).

Details

Summary of a Fitted Cross-Validated Parity Regression Model with Statistics

This function is an S3 method for the generic summary function. It formats and prints a detailed statistical overview of the optimal cross-validated model. The output includes:

- The parameterization method used ("budget" or "target").
- The matched call that produced the model.
- Residual quantiles.
- A table of estimated optimal coefficients with their corresponding standard errors, t-values, p-values, confidence intervals, and significance codes.
- Overall model fit statistics, including Residual Standard Error, Multiple and Adjusted R-squared, F-statistic, AIC, BIC, and Deviance.
- A Cross-Validation Summary displaying the minimum mean cross-validation error and the optimal tuning values (val and/or lambda).

Value

Invisibly returns NULL. This function is primarily called for its side effect of printing the summary to the console.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

cv.savvyPR

Examples

```
# Simulate some data
set.seed(123)
n <- 100
p <- 10
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5)

# Example 1: Fit and summarize a budget-based cross-validated model
cv_fit_budget <- cv.savvyPR(x, y, method = "budget", model_type = "PR3")
summary(cv_fit_budget)
```

```
# Example 2: Fit and summarize a target-based cross-validated model
cv_fit_target <- cv.savvyPR(x, y, method = "target", model_type = "PR1")
summary(cv_fit_target)
```

`summary.savvyPR` *Summary of Parity Regression Model*

Description

Prints a comprehensive statistical summary of a fitted parity regression model object. It displays the model's configuration, statistical summaries of the estimated coefficients, and overall model fit statistics.

Usage

```
## S3 method for class 'savvyPR'
summary(object, ...)
```

Arguments

`object`	A fitted model object of class `savvyPR` returned by `savvyPR`.
`...`	Additional arguments passed to the generic `summary` function (currently unused).

Details

Summary of a Fitted Parity Regression Model with Statistics

This function is an S3 method for the generic `summary` function. It formats and prints a detailed statistical overview of the fitted model. The output includes:

- The parameterization method used (`"budget"` or `"target"`).
- The matched call that produced the model.
- Residual quantiles.
- A table of estimated coefficients with their corresponding standard errors, t-values, p-values, confidence intervals, and significance codes.
- Overall model fit statistics, including Residual Standard Error, Multiple and Adjusted R-squared, F-statistic, AIC, BIC, and Deviance.

Value

Invisibly returns `NULL`. This function is primarily called for its side effect of printing the summary to the console.

Author(s)

Ziwei Chen, Vali Asimit and Pietro Millossovich
Maintainer: Ziwei Chen <ziwei.chen.3@citystgeorges.ac.uk>

See Also

savvyPR

Examples

```
# Simulate some data
set.seed(123)
n <- 100
p <- 10
x <- matrix(rnorm(n * p), n, p)
beta <- matrix(rnorm(p), p, 1)
y <- x %*% beta + rnorm(n, sd = 0.5)

# Example 1: Fit and summarize a Budget-based parity regression model
fit_budget <- savvyPR(x, y, method = "budget", val = 0.05, intercept = FALSE)
summary(fit_budget)

# Example 2: Fit and summarize a Target-based parity regression model
fit_target <- savvyPR(x, y, method = "target", val = 1, intercept = TRUE)
summary(fit_target)
```

Index