# Package 'balnet'

May 5, 2026

**Title** Pathwise Estimation of Covariate Balancing Propensity Scores

**Version** 0.0.2

**Description** Provides pathwise estimation of regularized logistic propensity score models using covariate balancing loss functions rather than maximum likelihood. Regularization paths are fit via the 'adelie' elastic-net solver with a 'glmnet'-like interface, yielding balancing weights that target covariate balance for the ATE and ATT.
For details, see Sverdrup & Hastie (2026) <doi:10.48550/arXiv.2602.18577>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**LinkingTo** Rcpp, RcppEigen

**SystemRequirements** C++17

**Imports** Rcpp, Matrix, methods

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown

**URL** https://github.com/erikcs/balnet

**BugReports** https://github.com/erikcs/balnet/issues

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Erik Sverdrup [aut, cre],
Trevor Hastie [aut],
James Yang [ctb] (Author of adelie_core C++ library (vendored in inst/include/))

**Maintainer** Erik Sverdrup <erik.sverdrup@monash.edu>

**Repository** CRAN

**Date/Publication** 2026-05-04 23:30:02 UTC

# Contents



---

`balnet` *Pathwise estimation of covariate balancing propensity scores.*

---

## Description

Fits regularized logistic regression models using covariate balancing loss functions, yielding balancing weights targeting the ATE, ATT, or treated/control means.

## Usage

```
balnet(
  X,
  W,
  target = c("ATE", "ATT", "treated", "control"),
  sample.weights = NULL,
  max.imbalance = NULL,
  nlambda = 100L,
  lambda.min.ratio = 0.01,
  lambda = NULL,
  penalty.factor = NULL,
  groups = NULL,
  alpha = 1,
  standardize = TRUE,
  tol = 1e-07,
  maxit = as.integer(1e+05),
  verbose = FALSE,
  num.threads = 1L,
  ...
)
```

## Arguments

| | |
|---|---|
| X | A numeric matrix or data frame with pre-treatment covariates. |
| W | Treatment vector (0 = control, 1 = treated). |
| target | The target estimand. Default is "ATE". |
| sample.weights | Optional sample weights. If NULL (default), each unit receives the same weight. |
| max.imbalance | Optional upper bound on the standardized covariate imbalance. For lasso penalization (alpha = 1), there is a one-to-one correspondence between the penalty parameter $\lambda$ and the maximum allowable covariate imbalance. When supplied, max.imbalance is used to adjust the lambda sequence (via lambda.min.ratio) so that the generated sequence ends at the specified imbalance level. |
| nlambda | Number of values for lambda if generated automatically. Default is 100. |
| lambda.min.ratio | Ratio of smallest to largest lambda. Default is 1e-2. |
| lambda | Optional lambda sequence. By default, it is constructed automatically using nlambda and lambda.min.ratio (or max.imbalance, if specified). |
| penalty.factor | Penalty factor per feature. Default is 1 (i.e., each feature receives the same penalty). |
| groups | Optional list of group indices for group penalization. |
| alpha | Elastic net mixing parameter. Default is 1 (lasso), 0 corresponds to ridge. For alpha = 0, the lambda sequence is constructed using a small positive alpha value (similar to glmnet), since $\lambda_{max} \to \infty$ as $\alpha \to 0$. |
| standardize | Whether to standardize the input matrix. Should only be FALSE if X already has zero-mean columns with unit variance. For target = "ATT", standardization should be based on the treated group. |
| tol | Coordinate descent convergence tolerance. Default is 1e-7. |
| maxit | Maximum number of coordinate descent iterations. Default is 1e5. |
| verbose | Whether to display information during fitting. Default is FALSE. |
| num.threads | Number of threads to use. Default is 1. |
| ... | Additional internal arguments passed to the solver. |

## Details

This function aims to find balancing weights $\hat{\gamma}_i$, using logistic propensity scores, that balance covariate means to a target vector, i.e.,

$$\frac{1}{n}\sum_{i=1}^{n}\hat{\gamma}_i X_i = \bar{X}_{\text{target}}.$$

With lasso regularization (alpha = 1), imbalance is controlled in the $\ell_\infty$ sense, allowing absolute slack of at most $\lambda$ per covariate.

For target = "ATE", two logistic models are fit, one per arm, with

$$\hat{\gamma}_i^{(1)} = \frac{W_i}{\hat{e}^{(1)}(X_i)}, \quad \hat{\gamma}_i^{(0)} = \frac{1-W_i}{1-\hat{e}^{(0)}(X_i)}, \quad \bar{X}_{\text{target}} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

$\hat{e}^{(w)}(X_i)$ is the fitted propensity score for arm $w$. For `target = "ATT"`, weights balance the control means:

$$\hat{\gamma}_i = (1 - W_i)\frac{\hat{e}^{(0)}(X_i)}{1 - \hat{e}^{(0)}(X_i)}, \quad \bar{X}_{\text{target}} = \frac{1}{\sum W_i}\sum_{i=1}^{n} W_i X_i.$$

## Value

A fit balnet object.

## References

Sverdrup, Erik and Trevor Hastie. "balnet: Pathwise Estimation of Covariate Balancing Propensity Scores". arXiv preprint, arXiv:2602.18577, 2026.

## Examples

```
# Simulate data with confounding.
n <- 2000
p <- 10
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1.5 + exp(X[, 2] + X[, 3])))
Y <- W + 2 * log(1 + exp(X[, 1] + X[, 2] + X[, 3])) + rnorm(n)

# Fit model targeting the ATE = E[Y(1)] - E[Y(0)].
# Two logistic models are fit: one for treated, one for control.
fit <- balnet(X, W, target = "ATE")

# Print path summary.
print(fit)

# Visualize the path.
plot(fit)

# Plot the standardized covariate imbalance at given lambda.
# Note: lambda = 0 selects the final lambda in the sequence. Scalar values
# are applied to both arms.
plot(fit, lambda = 0)

# Predict propensity scores at end of lambda path.
W.hat <- predict(fit, X, lambda = 0)

# Get balancing weights at end of lambda path.
ipw.weights <- balweights(fit, lambda = 0)

# Estimate ATE using balancing weights.
mean(Y * (ipw.weights$treated - ipw.weights$control))
```

`balweights` *Extract balancing weights from a balnet object.*

## Description

Retrieves the estimated balancing weights $\hat{\gamma}$. Under unconfoundedness, these correspond to inverse probability weights (IPW) for standard treatment effect estimands.

## Usage

```
balweights(object, lambda = NULL, ...)

## S3 method for class 'balnet'
balweights(object, lambda = NULL, ...)

## S3 method for class 'cv.balnet'
balweights(object, lambda = "lambda.min", ...)
```

## Arguments

| | |
|---|---|
| `object` | A `balnet` object. |
| `lambda` | Value(s) of the penalty parameter `lambda` at which weights are required.<br>• If `NULL` (default), the full lambda path from the fit is used.<br>• If new values are supplied, linear interpolation is performed. For dual-arm fits (`target` = "ATE"), `lambda` can be a `list` or two-column `matrix`: the first element/column corresponds to the control arm and the second to the treatment. |
| `...` | Additional arguments (currently ignored). |

## Value

Estimated balancing weights (for contrast fits, `target` = "ATE" or "ATT", returns a list with entries for each arm).

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
fit <- balnet(X, W, target = "ATT")

# Extract balancing weights.
wts <- balweights(fit, lambda = 0)
```

coef.balnet *Extract coefficients from a balnet object.*

## Description

Extract coefficients from a balnet object.

## Usage

```
## S3 method for class 'balnet'
coef(object, lambda = NULL, ...)
```

## Arguments

| | |
|---|---|
| object | A balnet object. |
| lambda | Value(s) of the penalty parameter lambda at which coefficients are required.<br>• If NULL (default), the full lambda path from the fit is used.<br>• If new values are supplied, linear interpolation is performed. For dual-arm fits (target = "ATE"), lambda can be a list or two-column matrix: the first element/column corresponds to the control arm and the second to the treatment. |
| ... | Additional arguments (currently ignored). |

## Value

Estimated logistic coefficients (for dual-arm fits, returns a list with entries for each arm).

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
fit <- balnet(X, W, target = "ATT")

# Extract coefficients.
coefs <- coef(fit)
```

`coef.cv.balnet` *Extract coefficients from a cv.balnet object.*

## Description

Extract coefficients from a cv.balnet object.

## Usage

```
## S3 method for class 'cv.balnet'
coef(object, lambda = "lambda.min", ...)
```

## Arguments

| | |
|---|---|
| `object` | A `cv.balnet` object. |
| `lambda` | The lambda to use. Defaults to the cross-validated lambda. |
| `...` | Additional arguments (currently ignored). |

## Value

Estimated logistic coefficients (for dual-arm fits, returns a list with entries for each arm).

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
cv.fit <- cv.balnet(X, W, target = "ATT")

# Extract coefficients at cross-validated lambda.
coefs <- coef(cv.fit)
```

`cv.balnet` *Cross-validation for balnet.*

## Description

Cross-validation for balnet.

## Usage

```
cv.balnet(
  X,
  W,
  type.measure = c("balance.loss"),
  nfolds = 10,
  foldid = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| X | A numeric matrix or data frame with pre-treatment covariates. |
| W | Treatment vector (0: control, 1: treated). |
| type.measure | The loss to minimize for cross-validation. Default is balance loss. |
| nfolds | The number of folds used for cross-validation, default is 10. |
| foldid | An optional n-vector specifying which fold 1 to nfold a sample belongs to. If NULL, this defaults to sample(rep(seq(nfolds), length.out = nrow(X))). |
| ... | Arguments for balnet. |

## Value

A fit cv.balnet object.

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATE model.
cv.fit <- cv.balnet(X, W)

# Print CV summary.
print(cv.fit)

# Plot at cross-validated lambda.
plot(cv.fit)

# Predict at cross-validated lambda.
W.hat <- predict(cv.fit, X)
```

plot.balnet *Plot diagnostics for a* balnet *object.*

## Description

Shows effective sample size (ESS) and percent bias reduction (PBR; reduction in mean absolute imbalance) along the regularization path, computed from balancing weights and normalized to percentages. The right-hand axis maps these values to the coefficient of variation (CV) of the weights. Supplying the lambda argument displays the standardized covariate imbalance $(\bar{X}_{\text{weighted}} - \bar{X}_{\text{target}})/\sigma_{\text{target}}$, computed using the balancing weights at the specified lambda.

## Usage

```
## S3 method for class 'balnet'
plot(x, lambda = NULL, groups = NULL, max = NULL, ...)
```

## Arguments

| | |
|---|---|
| x | A balnet object. |
| lambda | If NULL (default) diagnostics over the lambda path is shown. Otherwise, covariate balance at provided lambda value is shown (if target = "ATE", lambda can be a 2-vector, arm 0 and arm 1.) |
| groups | Optional named list of contiguous covariate index ranges to aggregate into a single variable before computing covariate imbalance (e.g., list(demographics = 4:12)). |
| max | The number of covariates to display in covariate balance plot. Defaults to all covariates. |
| ... | Additional arguments. |

## Value

Invisibly returns the information underlying the plot.

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
fit <- balnet(X, W, target = "ATT")

# Plot the five covariates with the largest unweighted imbalance
plot(fit, lambda = 0, max = 5)
```

## plot.cv.balnet *Plot diagnostics for a* cv.balnet *object.*

### Description

Plot diagnostics for a cv.balnet object.

### Usage

```
## S3 method for class 'cv.balnet'
plot(x, lambda = "lambda.min", ...)
```

### Arguments

| | |
|---|---|
| x | A cv.balnet object. |
| lambda | The lambda to use. Defaults to the cross-validated lambda. |
| ... | Additional arguments. |

### Value

Invisibly returns the information underlying the plot.

### Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
cv.fit <- cv.balnet(X, W, target = "ATT")

# Plot at cross-validated lambda.
plot(cv.fit)
```

## predict.balnet *Predict using a balnet object.*

### Description

Predict using a balnet object.

## Usage

```
## S3 method for class 'balnet'
predict(object, newdata, lambda = NULL, type = c("response"), ...)
```

## Arguments

| | |
|---|---|
| object | A balnet object. |
| newdata | A numeric matrix. |
| lambda | Value(s) of the penalty parameter lambda at which coefficients are required.<br>• If NULL (default), the full lambda path from the fit is used.<br>• If new values are supplied, linear interpolation is performed. For dual-arm fits (target = "ATE"), lambda can be a list or two-column matrix: the first element/column corresponds to the control arm and the second to the treatment. |
| type | The type of predictions. Default is "response" (propensity scores). |
| ... | Additional arguments (currently ignored). |

## Value

Estimated predictions (for dual-arm fits, returns a list with entries for each arm).

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
fit <- balnet(X, W, target = "ATT")

# Predict propensity scores.
W.hat <- predict(fit, X)
```

---

# predict.cv.balnet *Predict using a cv.balnet object.*

---

## Description

Predict using a cv.balnet object.

## Usage

```
## S3 method for class 'cv.balnet'
predict(object, newdata, lambda = "lambda.min", type = c("response"), ...)
```

### Arguments

| | |
|---|---|
| object | A cv.balnet object. |
| newdata | A numeric matrix. |
| lambda | The lambda to use. Defaults to the cross-validated lambda. |
| type | The type of predictions. Default is "response" (propensity scores). |
| ... | Additional arguments (currently ignored). |

### Value

Estimated predictions (for dual-arm fits, returns a list with entries for each arm).

### Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
cv.fit <- cv.balnet(X, W, target = "ATT")

# Predict propensity scores at cross-validated lambda.
W.hat <- predict(cv.fit, X)
```

---

## print.balnet *Print a balnet object.*

---

### Description

Print a balnet object.

### Usage

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

### Arguments

| | |
|---|---|
| x | A balnet object. |
| digits | Number of digits to print. |
| max | Total number of rows to show from the beginning and end of the path |
| ... | Additional print arguments. |

## Value

Invisibly returns the printed information.

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
fit <- balnet(X, W, target = "ATT")

# Print path summary.
print(fit)
```

---

`print.cv.balnet` *Print a cv.balnet object.*

---

## Description

Print a cv.balnet object.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | A `cv.balnet` object. |
| `digits` | Number of digits to print. |
| `...` | Additional print arguments. |

## Value

Invisibly returns the printed information.

## Examples

```
n <- 100
p <- 25
X <- matrix(rnorm(n * p), n, p)
W <- rbinom(n, 1, 1 / (1 + exp(1 - X[, 1])))

# Fit an ATT model.
```

```
cv.fit <- cv.balnet(X, W, target = "ATT")

# Print CV summary.
print(cv.fit)
```

# Index