

# Package ‘cornet’

July 22, 2025

**Version** 1.0.0

**Title** Penalised Regression for Dichotomised Outcomes

**Description** Implements lasso and ridge regression for dichotomised outcomes (<[doi:10.1080/02664763.2023.2233057](https://doi.org/10.1080/02664763.2023.2233057)>), i.e., numerical outcomes that were transformed to binary outcomes. Such artificial binary outcomes indicate whether an underlying measurement is greater than a threshold.

**Depends** R (>= 3.0.0)

**Imports** glmnet, palasso

**Suggests** knitr, testthat, rmarkdown, RColorBrewer, MASS, mvtnorm, randomForest, xgboost, MLmetrics

**License** GPL-3

**Encoding** UTF-8

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**URL** <https://github.com/rauschenberger/cornet>,  
<https://rauschenberger.github.io/cornet/>

**BugReports** <https://github.com/rauschenberger/cornet/issues>

**NeedsCompilation** no

**Author** Armin Rauschenberger [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-6498-4801>>)

**Maintainer** Armin Rauschenberger <[armin.rauschenberger@uni.lu](mailto:armin.rauschenberger@uni.lu)>

**Repository** CRAN

**Date/Publication** 2024-09-26 22:50:07 UTC

## Contents

|                     |   |
|---------------------|---|
| .check . . . . .    | 2 |
| .equal . . . . .    | 3 |
| .simulate . . . . . | 3 |

|                          |    |
|--------------------------|----|
| .test . . . . .          | 4  |
| coef.cornet . . . . .    | 5  |
| cornet . . . . .         | 5  |
| cv.cornet . . . . .      | 7  |
| plot.cornet . . . . .    | 9  |
| predict.cornet . . . . . | 9  |
| print.cornet . . . . .   | 10 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>12</b> |
|--------------|-----------|

---

|               |                  |
|---------------|------------------|
| <i>.check</i> | <i>Arguments</i> |
|---------------|------------------|

---

**Description**

Verifies whether an argument matches formal requirements.

**Usage**

```
.check(  
  x,  
  type,  
  dim = NULL,  
  miss = FALSE,  
  min = NULL,  
  max = NULL,  
  values = NULL,  
  inf = FALSE,  
  null = FALSE  
)
```

**Arguments**

|        |                                                     |
|--------|-----------------------------------------------------|
| x      | argument                                            |
| type   | character "string", "scalar", "vector", "matrix"    |
| dim    | vector/matrix dimensionality: integer scalar/vector |
| miss   | accept missing values: logical                      |
| min    | lower limit: numeric                                |
| max    | upper limit: numeric                                |
| values | only accept specific values: vector                 |
| inf    | accept infinite (Inf or -Inf) values: logical       |
| null   | accept NULL: logical                                |

**Examples**

```
cornet:::.check(0.5, type="scalar", min=0, max=1)
```

---

|               |                 |
|---------------|-----------------|
| <i>.equal</i> | <i>Equality</i> |
|---------------|-----------------|

---

**Description**

Verifies whether two or more arguments are identical.

**Usage**

```
.equal(..., na.rm = FALSE)
```

**Arguments**

|       |                                                   |
|-------|---------------------------------------------------|
| ...   | scalars, vectors, or matrices of equal dimensions |
| na.rm | remove missing values: logical                    |

**Examples**

```
cornet:::.equal(1,1,1)
```

---

|                  |                        |
|------------------|------------------------|
| <i>.simulate</i> | <i>Data simulation</i> |
|------------------|------------------------|

---

**Description**

Simulates data for unit tests

**Usage**

```
.simulate(n, p, cor = 0, prob = 0.1, sd = 1, exp = 1, frac = 1)
```

**Arguments**

|      |                                                   |
|------|---------------------------------------------------|
| n    | sample size: positive integer                     |
| p    | covariate space: positive integer                 |
| cor  | correlation coefficient : numeric between 0 and 1 |
| prob | effect proportion: numeric between 0 and 1        |
| sd   | standard deviation: positive numeric              |
| exp  | exponent: positive numeric                        |
| frac | class proportion: numeric between 0 and 1         |

Details

For simulating correlated features ( $\text{cor} > 0$ ), this function requires the R package MASS (see [mvnrm](#)).

Value

Returns invisible list with elements `y` and `X`.

Examples

```
data <- cornet:::simulate(n=10,p=20)
names(data)
```

---

|                    |                          |
|--------------------|--------------------------|
| <code>.test</code> | <i>Single-split test</i> |
|--------------------|--------------------------|

---

Description

Compares models for a continuous response with a cut-off value.

Usage

```
.test(y, cutoff, X, alpha = 1, type.measure = "deviance")
```

Arguments

|                           |                                                                                                                          |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>y</code>            | continuous outcome: vector of length $n$                                                                                 |
| <code>cutoff</code>       | cut-off point for dichotomising outcome into classes: <i>meaningful</i> value between $\min(y)$ and $\max(y)$            |
| <code>X</code>            | features: numeric matrix with $n$ rows (samples) and $p$ columns (variables)                                             |
| <code>alpha</code>        | elastic net mixing parameter: numeric between 0 (ridge) and 1 (lasso)                                                    |
| <code>type.measure</code> | loss function for binary classification: character "deviance", "mse", "mae", or "class" (see <a href="#">cv.glmnet</a> ) |

Details

Splits samples into 80 percent for training and 20 percent for testing, calculates squared deviance residuals of logistic and combined regression, conducts the paired one-sided Wilcoxon signed rank test, and returns the  $p$ -value. For the multi-split test, use the median  $p$ -value from 50 single-split tests (van de Wiel 2009).

Examples

```
n <- 100; p <- 200
y <- rnorm(n)
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
cornet:::test(y=y,cutoff=0,X=X)
```

---

|             |                                       |
|-------------|---------------------------------------|
| coef.cornet | <i>Extract estimated coefficients</i> |
|-------------|---------------------------------------|

---

### Description

Extracts estimated coefficients from linear and logistic regression, under the penalty parameter that minimises the cross-validated loss.

### Usage

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

### Arguments

|        |                                    |
|--------|------------------------------------|
| object | <a href="#">cornet</a> object      |
| ...    | further arguments (not applicable) |

### Value

This function returns a matrix with  $n$  rows and two columns, where  $n$  is the sample size. It includes the estimated coefficients from linear regression (1st column: "beta") and logistic regression (2nd column: "gamma").

### Examples

```
n <- 100; p <- 200
y <- rnorm(n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
net <- cornet(y=y, cutoff=0, X=X)
coef(net)
```

---

|        |                            |
|--------|----------------------------|
| cornet | <i>Combined regression</i> |
|--------|----------------------------|

---

### Description

Implements lasso and ridge regression for dichotomised outcomes. Such outcomes are not naturally but artificially binary. They indicate whether an underlying measurement is greater than a threshold.

**Usage**

```
cornet(
  y,
  cutoff,
  X,
  alpha = 1,
  np_i = 101,
  pi = NULL,
  nsigma = 99,
  sigma = NULL,
  nfolds = 10,
  foldid = NULL,
  type.measure = "deviance",
  ...
)
```

**Arguments**

|                           |                                                                                                                                   |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <code>y</code>            | continuous outcome: vector of length $n$                                                                                          |
| <code>cutoff</code>       | cut-off point for dichotomising outcome into classes: <i>meaningful</i> value between <code>min(y)</code> and <code>max(y)</code> |
| <code>X</code>            | features: numeric matrix with $n$ rows (samples) and $p$ columns (variables)                                                      |
| <code>alpha</code>        | elastic net mixing parameter: numeric between 0 (ridge) and 1 (lasso)                                                             |
| <code>np_i</code>         | number of <code>pi</code> values (weighting)                                                                                      |
| <code>pi</code>           | <code>pi</code> sequence: vector of increasing values in the unit interval; or <code>NULL</code> (default sequence)               |
| <code>nsigma</code>       | number of <code>sigma</code> values (scaling)                                                                                     |
| <code>sigma</code>        | <code>sigma</code> sequence: vector of increasing positive values; or <code>NULL</code> (default sequence)                        |
| <code>nfolds</code>       | number of folds: integer between 3 and $n$                                                                                        |
| <code>foldid</code>       | fold identifiers: vector with entries between 1 and <code>nfolds</code> ; or <code>NULL</code> (balance)                          |
| <code>type.measure</code> | loss function for binary classification: character "deviance", "mse", "mae", or "class" (see <a href="#">cv.glmnet</a> )          |
| <code>...</code>          | further arguments passed to <a href="#">glmnet</a>                                                                                |

**Details**

The argument `family` is unavailable, because this function fits a *gaussian* model for the numeric response, and a *binomial* model for the binary response.

Linear regression uses the loss function "deviance" (or "mse"), but the loss is incomparable between linear and logistic regression.

The loss function "auc" is unavailable for internal cross-validation. If at all, use "auc" for external cross-validation only.

**Value**

Returns an object of class `cornet`, a list with multiple slots:

- `gaussian`: fitted linear model, class `glmnet`
- `binomial`: fitted logistic model, class `glmnet`
- `sigma`: scaling parameters `sigma`, vector of length `nsigma`
- `pi`: weighting parameters `pi`, vector of length `npi`
- `cvm`: evaluation loss, matrix with `nsigma` rows and `npi` columns
- `sigma.min`: optimal scaling parameter, positive scalar
- `pi.min`: optimal weighting parameter, scalar in unit interval
- `cutoff`: threshold for dichotomisation

**References**

Armin Rauschenberger and Enrico Glaab (2024). "Predicting dichotomised outcomes from high-dimensional data in biomedicine". *Journal of Applied Statistics* 51(9):1756-1771. doi:10.1080/02664763.2023.2233057. (Click [here](#) to access PDF. Contact: <armin.rauschenberger@uni.lu>.)

**See Also**

Methods for objects of class `cornet` include [coef](#) and [predict](#).

**Examples**

```
n <- 100; p <- 200
y <- rnorm(n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
net <- cornet(y=y, cutoff=0, X=X)
net
```

---

cv.cornet

---

*Performance measurement*


---

**Description**

Compares models for a continuous response with a cut-off value.

**Usage**

```
cv.cornet(
  y,
  cutoff,
  X,
  alpha = 1,
  nfold.ext = 5,
```

```

    nfolds.int = 10,
    foldid.ext = NULL,
    foldid.int = NULL,
    type.measure = "deviance",
    rf = FALSE,
    xgboost = FALSE,
    ...
)

```

## Arguments

|                           |                                                                                                                          |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>y</code>            | continuous outcome: vector of length $n$                                                                                 |
| <code>cutoff</code>       | cut-off point for dichotomising outcome into classes: <i>meaningful</i> value between $\min(y)$ and $\max(y)$            |
| <code>X</code>            | features: numeric matrix with $n$ rows (samples) and $p$ columns (variables)                                             |
| <code>alpha</code>        | elastic net mixing parameter: numeric between 0 (ridge) and 1 (lasso)                                                    |
| <code>nfolds.ext</code>   | number of external folds                                                                                                 |
| <code>nfolds.int</code>   | internal fold identifiers: vector of length $n$ with entries between 1 and <code>nfolds.int</code> ; or NULL             |
| <code>foldid.ext</code>   | external fold identifiers: vector of length $n$ with entries between 1 and <code>nfolds.ext</code> ; or NULL             |
| <code>foldid.int</code>   | number of internal folds                                                                                                 |
| <code>type.measure</code> | loss function for binary classification: character "deviance", "mse", "mae", or "class" (see <a href="#">cv.glmnet</a> ) |
| <code>rf</code>           | comparison with random forest: logical                                                                                   |
| <code>xgboost</code>      | comparison with extreme gradient boosting: logical                                                                       |
| <code>...</code>          | further arguments passed to <a href="#">cornet</a> or <a href="#">glmnet</a>                                             |

## Details

Computes the cross-validated loss of logistic and combined regression.

## Examples

```

## Not run: n <- 100; p <- 200
y <- rnorm(n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
start <- Sys.time()
loss <- cv.cornet(y=y, cutoff=0, X=X)
end <- Sys.time()
end - start

loss
## End(Not run)

```

---

plot.cornet

*Plot loss matrix*


---

### Description

Plots the loss for different combinations of scaling (sigma) and weighting (pi) parameters.

### Usage

```
## S3 method for class 'cornet'
plot(x, ...)
```

### Arguments

x cornet object  
 ... further arguments (not applicable)

### Value

This function plots the evaluation loss (cvm). Whereas the matrix has sigma in the rows, and pi in the columns, the plot has sigma on the *x*-axis, and pi on the *y*-axis. For all combinations of sigma and pi, the colour indicates the loss. If the R package RColorBrewer is installed, blue represents low. Otherwise, red represents low. White always represents high.

### Examples

```
n <- 100; p <- 200
y <- rnorm(n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
net <- cornet(y=y, cutoff=0, X=X)
plot(net)
```

---

predict.cornet

*Predict binary outcome*


---

### Description

Predicts the binary outcome with linear, logistic, and combined regression.

### Usage

```
## S3 method for class 'cornet'
predict(object, newx, type = "probability", ...)
```

**Arguments**

|        |                                                                                |
|--------|--------------------------------------------------------------------------------|
| object | <code>cornet</code> object                                                     |
| newx   | covariates: numeric matrix with $n$ rows (samples) and $p$ columns (variables) |
| type   | "probability", "odds", "log-odds"                                              |
| ...    | further arguments (not applicable)                                             |

**Details**

For linear regression, this function tentatively transforms the predicted values to predicted probabilities, using a Gaussian distribution with a fixed mean (threshold) and a fixed variance (estimated variance of the numeric outcome).

**Examples**

```
n <- 100; p <- 200
y <- rnorm(n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
net <- cornet(y=y, cutoff=0, X=X)
predict(net, newx=X)
```

---

|              |                            |
|--------------|----------------------------|
| print.cornet | <i>Combined regression</i> |
|--------------|----------------------------|

---

**Description**

Prints summary of cornet object.

**Usage**

```
## S3 method for class 'cornet'
print(x, ...)
```

**Arguments**

|     |                                    |
|-----|------------------------------------|
| x   | <code>cornet</code> object         |
| ... | further arguments (not applicable) |

**Value**

Returns sample size  $n$ , number of covariates  $p$ , information on dichotomisation, tuned scaling parameter (sigma), tuned weighting parameter (pi), and corresponding loss.

**Examples**

```
n <- 100; p <- 200
y <- rnorm(n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
net <- cornet(y=y, cutoff=0, X=X)
print(net)
```

# Index

`.check`, [2](#)  
`.equal`, [3](#)  
`.simulate`, [3](#)  
`.test`, [4](#)  
  
`coef`, [7](#)  
`coef.cornet`, [5](#)  
`cornet`, [5](#), [5](#), [8–10](#)  
`cornet-package (cornet)`, [5](#)  
`cv.cornet`, [7](#)  
`cv.glmnet`, [4](#), [6](#), [8](#)  
  
`glmnet`, [6](#), [8](#)  
  
`mvnrm`, [4](#)  
  
`plot.cornet`, [9](#)  
`predict`, [7](#)  
`predict.cornet`, [9](#)  
`print.cornet`, [10](#)