

# Package: starnet (via r-universe)

September 3, 2024

**Version** 0.0.6

**Title** Stacked Elastic Net

**Description** Implements stacked elastic net regression (Rauschenberger 2020, <[doi:10.1093/bioinformatics/btaa535](https://doi.org/10.1093/bioinformatics/btaa535)>). The elastic net generalises ridge and lasso regularisation (Zou 2005, <[doi:10.1111/j.1467-9868.2005.00503.x](https://doi.org/10.1111/j.1467-9868.2005.00503.x)>). Instead of fixing or tuning the mixing parameter alpha, we combine multiple alpha by stacked generalisation (Wolpert 1992 <[doi:10.1016/S0893-6080\(05\)80023-1](https://doi.org/10.1016/S0893-6080(05)80023-1)>).

**Depends** R (>= 3.0.0)

**Imports** glmnet, survival, cornet, Matrix

**Suggests** knitr, testthat, rmarkdown

**Enhances** CVXR, mvtnorm

**VignetteBuilder** knitr

**License** GPL-3

**LazyData** true

**Language** en-GB

**RoxygenNote** 7.1.1

**URL** <https://github.com/rauschenberger/starnet>

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

**Repository** <https://rauschenberger.r-universe.dev>

**RemoteUrl** <https://github.com/rauschenberger/starnet>

**RemoteRef** HEAD

**RemoteSha** ff142156df69ff48ffe4b9755b6a8ec20037345f

## Contents

starnet-package . . . . .	2
.cv.glmnet . . . . .	2

.loss . . . . .	3
.simulate . . . . .	4
coef.starnet . . . . .	4
cv.starnet . . . . .	5
glmnet.auc . . . . .	6
predict.starnet . . . . .	7
print.starnet . . . . .	8
starnet . . . . .	9
weights.starnet . . . . .	10

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

---

starnet-package	<i>Stacked Elastic Net Regression</i>
-----------------	---------------------------------------

---

## Description

The R package `starnet` implements stacked elastic net regression. The elastic net generalises ridge and lasso regularisation. Instead of fixing or tuning the mixing parameter  $\alpha$ , we combine multiple alphas by stacked generalisation.

## Details

Use function `starnet` for model fitting. Type `library(starnet)` and then `?starnet` or `help("starnet")` to open its help file.

See the vignette for further examples. Type `vignette("starnet")` or `browseVignettes("starnet")` to open the vignette.

## References

A Rauschenberger, E Glaab, and MA van de Wiel (2020). "Predictive and interpretable models via the stacked elastic net". *Bioinformatics*. In press. doi:10.1093/bioinformatics/btaa535. <armin.rauschenberger@uni.lu>

---

.cv.glmnet	<i>glmnet::cv.glmnet</i>
------------	--------------------------

---

## Description

Wrapper for `cv.glmnet`, with different handling of sparsity constraints.

## Usage

```
.cv.glmnet(..., nzero)
```

**Arguments**

... see [cv.glmnet](#)  
 nzero maximum number of non-zero coefficients: positive integer

**Value**

Object of class [cv.glmnet](#).

**Examples**

NA

---

.loss	<i>Loss</i>
-------	-------------

---

**Description**

Calculate loss from predicted and observed values

**Usage**

`.loss(y, x, family, type.measure, foldid = NULL, grouped = TRUE)`

**Arguments**

y observed values: numeric vector of length  $n$   
 x predicted values: numeric vector of length  $n$   
 family character "gaussian", "binomial", "poisson", "mgaussian", or "multinomial"  
 (to implement: "cox")  
 type.measure character "deviance", "mse", "mae", "class", or "auc"  
 foldid fold identifiers: integer vector of length  $n$ , or NULL  
 grouped logical (for "cox" only)

**Examples**

NA

---

`.simulate`*Simulation*

---

**Description**

Functions for simulating data

**Usage**

```
.simulate.block(n, p, mode, family = "gaussian")
```

**Arguments**

n	sample size: positive integer
p	dimensionality: positive integer
mode	character "sparse", "dense" or "mixed"
family	character "gaussian", "binomial" or "poisson"

**Value**

List of vector y and matrix X.

**Examples**

```
NA
```

---

`coef.starnet`*Extract Coefficients*

---

**Description**

Extracts pooled coefficients. (The meta learners weights the coefficients from the base learners.)

**Usage**

```
## S3 method for class 'starnet'
coef(object, nzero = NULL, ...)
```

**Arguments**

object	<a href="#">starnet</a> object
nzero	maximum number of non-zero coefficients: positive integer, or NULL
...	further arguments (not applicable)

**Value**

List of scalar alpha and vector beta, containing the pooled intercept and the pooled slopes, respectively.

**Examples**

```
set.seed(1)
n <- 50; p <- 100
y <- rnorm(n=n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
object <- starnet(y=y, X=X)
coef <- coef(object)
```

---

cv.starnet

*Model comparison*

---

**Description**

Compares stacked elastic net, tuned elastic net, ridge and lasso.

**Usage**

```
cv.starnet(
  y,
  X,
  family = "gaussian",
  nalpha = 21,
  alpha = NULL,
  nfolds.ext = 10,
  nfolds.int = 10,
  foldid.ext = NULL,
  foldid.int = NULL,
  type.measure = "deviance",
  alpha.meta = 1,
  nzero = NULL,
  intercept = NULL,
  upper.limit = NULL,
  unit.sum = NULL,
  ...
)
```

**Arguments**

**y** response: numeric vector of length  $n$   
**X** covariates: numeric matrix with  $n$  rows (samples) and  $p$  columns (variables)

family	character "gaussian", "binomial" or "poisson"
nalpha	number of alpha values
alpha	elastic net mixing parameters: vector of length nalpha with entries between 0 (ridge) and 1 (lasso); or NULL (equidistance)
nfolds.ext, nfolds.int, foldid.ext, foldid.int	number of folds (nfolds): positive integer; fold identifiers (foldid): vector of length $n$ with entries between 1 and nfolds, or NULL, for hold-out (single split) instead of cross-validation (multiple splits): set foldid.ext to 0 for training and to 1 for testing samples
type.measure	loss function: character "deviance", "class", "mse" or "mae" (see <a href="#">cv.glmnet</a> )
alpha.meta	meta-learner: value between 0 (ridge) and 1 (lasso) for elastic net regularisation; NA for convex combination
nzero	number of non-zero coefficients: scalar/vector including positive integer(s) or NA; or NULL (no post hoc feature selection)
intercept	settings for meta-learner: logical, or NULL (intercept=!is.na(alpha.meta), upper.limit=TRUE, unit.sum=is.na(alpha.meta))
upper.limit	settings for meta-learner: logical, or NULL (intercept=!is.na(alpha.meta), upper.limit=TRUE, unit.sum=is.na(alpha.meta))
unit.sum	settings for meta-learner: logical, or NULL (intercept=!is.na(alpha.meta), upper.limit=TRUE, unit.sum=is.na(alpha.meta))
...	further arguments (not applicable)

### Value

List containing the cross-validated loss (or out-of sample loss if `nfolds.ext` equals two, and `foldid.ext` contains zeros and ones). The slot `meta` contains the loss from the stacked elastic net (`stack`), the tuned elastic net (`tune`), ridge, lasso, and the intercept-only model (`none`). The slot `base` contains the loss from the base learners. And the slot `extra` contains the loss from the restricted stacked elastic net (`stack`), lasso, and lasso-like elastic net (`enet`), with the maximum number of non-zero coefficients shown in the column name.

### Examples

```
loss <- cv.starnet(y=y, X=X)
```

---

glmnet.auc

*glmnet:::auc*

---

### Description

Import of [auc](#) (internal function)

**Usage**

```
glmnet.auc(y, prob, w)
```

**Arguments**

y	observed classes
prob	predicted probabilities
w	(ignored here)

**Value**

area under the ROC curve

**Examples**

```
NA
```

---

predict.starnet	<i>Makes Predictions</i>
-----------------	--------------------------

---

**Description**

Predicts outcome from features with stacked model.

**Usage**

```
## S3 method for class 'starnet'
predict(object, newx, type = "response", nzero = NULL, ...)
```

**Arguments**

object	<a href="#">starnet</a> object
newx	covariates: numeric matrix with $n$ rows (samples) and $p$ columns (variables)
type	character "link" or "response"
nzero	maximum number of non-zero coefficients: positive integer, or NULL
...	further arguments (not applicable)

**Value**

Matrix of predicted values, with samples in the rows, and models in the columns. Included models are alpha (fixed elastic net), ridge (i.e. alpha0), lasso (i.e. alpha1), tune (tuned elastic net), stack (stacked elastic net), and none (intercept-only model).

## Examples

```
set.seed(1)
n <- 50; p <- 100
y <- rnorm(n=n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
object <- starnet(y=y, X=X)
y_hat <- predict(object, newx=X[c(1),, drop=FALSE])
```

---

print.starnet

*Print Values*

---

## Description

Prints object of class [starnet](#).

## Usage

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

## Arguments

x                    [starnet](#) object  
...                   further arguments (not applicable)

## Value

Prints "stacked gaussian/binomial/poisson elastic net".

## Examples

```
set.seed(1)
n <- 50; p <- 100
y <- rnorm(n=n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
object <- starnet(y=y, X=X)
print(object)
```



---

starnet	<i>Stacked Elastic Net Regression</i>
---------	---------------------------------------

---

### Description

Implements stacked elastic net regression.

### Usage

```
starnet(
  y,
  X,
  family = "gaussian",
  nalpha = 21,
  alpha = NULL,
  nfolds = 10,
  foldid = NULL,
  type.measure = "deviance",
  alpha.meta = 1,
  penalty.factor = NULL,
  intercept = NULL,
  upper.limit = NULL,
  unit.sum = NULL,
  ...
)
```

### Arguments

<code>y</code>	response: numeric vector of length $n$
<code>X</code>	covariates: numeric matrix with $n$ rows (samples) and $p$ columns (variables)
<code>family</code>	character "gaussian", "binomial" or "poisson"
<code>nalpha</code>	number of alpha values
<code>alpha</code>	elastic net mixing parameters: vector of length <code>nalpha</code> with entries between 0 (ridge) and 1 (lasso); or NULL (equidistance)
<code>nfolds</code>	number of folds
<code>foldid</code>	fold identifiers: vector of length $n$ with entries between 1 and <code>nfolds</code> ; or NULL (balance)
<code>type.measure</code>	loss function: character "deviance", "class", "mse" or "mae" (see <a href="#">cv.glmnet</a> )
<code>alpha.meta</code>	meta-learner: value between 0 (ridge) and 1 (lasso) for elastic net regularisation; NA for convex combination
<code>penalty.factor</code>	differential shrinkage: vector of length $n$ with entries between 0 (include) and $Inf$ (exclude), or NULL (all 1)
<code>intercept</code> , <code>upper.limit</code> , <code>unit.sum</code>	settings for meta-learner: logical, or NULL ( <code>intercept=!is.na(alpha.meta)</code> , <code>upper.limit=TRUE</code> , <code>unit.sum=is.na(alpha.meta)</code> )
<code>...</code>	further arguments passed to <a href="#">glmnet</a>

**Details**

Post hoc feature selection: consider argument `nzero` in functions `coef` and `predict`.

**Value**

Object of class `starnet`. The slots `base` and `meta` contain `cv.glmnet`-like objects, for the base and meta learners, respectively.

**References**

A Rauschenberger, E Glaab, and MA van de Wiel (2020). "Predictive and interpretable models via the stacked elastic net". *Bioinformatics*. In press. doi:10.1093/bioinformatics/btaa535. <armin.rauschenberger@uni.lu>

**Examples**

```
set.seed(1)
n <- 50; p <- 100
y <- rnorm(n=n)
X <- matrix(rnorm(n*p), nrow=n, ncol=p)
object <- starnet(y=y, X=X, family="gaussian")
```

---

weights.starnet

*Extract Weights*


---

**Description**

Extracts coefficients from the meta learner, i.e. the weights for the base learners.

**Usage**

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

**Arguments**

```
object      starnet object
...         further arguments (not applicable)
```

**Value**

Vector containing intercept and slopes from the meta learner.

**Examples**

```
set.seed(1)
n <- 50; p <- 100
y <- rnorm(n=n)
X <- matrix(rnorm(n*p),nrow=n,ncol=p)
object <- starnet(y=y,X=X)
weights(object)
```

# Index

## \* **documentation**

starnet-package, [2](#)

.cv.glmnet, [2](#)

.loss, [3](#)

.simulate, [4](#)

auc, [6](#)

coef, [10](#)

coef.starnet, [4](#)

cv.glmnet, [2](#), [3](#), [6](#), [9](#), [10](#)

cv.starnet, [5](#)

glmnet, [9](#)

glmnet.auc, [6](#)

predict, [10](#)

predict.starnet, [7](#)

print.starnet, [8](#)

starnet, [2](#), [4](#), [7](#), [8](#), [9](#), [10](#)

starnet-package, [2](#)

weights.starnet, [10](#)