

# Package ‘transreg’

July 27, 2025

**Version** 1.0.5

**Title** Penalised Regression with Multiple Sets of Prior Effects  
(‘Transfer Learning’)

**Description** Improves the predictive performance of ridge and lasso regression exploiting one or more sources of prior information on the importance and direction of effects (Rauschenberger and others 2023, <[doi:10.1093/bioinformatics/btad680](https://doi.org/10.1093/bioinformatics/btad680)>). For running the vignette (optional), install ‘fwelnet’ and ‘ecpc’ from <<https://github.com/kjytay/fwelnet>> and <<https://github.com/Mirrelijn/ecpc>>, respectively.

**Depends** R (>= 3.0.0)

**Imports** glmnet, starnet, joint

**Suggests** knitr, testthat, rmarkdown, markdown, mvtnorm, remotes,  
glmtrans, xrnet, ecpc, fwelnet, doMC, palasso, xtable,  
devtools, CVXR

**License** GPL-3

**Encoding** UTF-8

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

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

**BugReports** <https://github.com/rauschenberger/transreg/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** 2025-07-26 22:40:02 UTC

## Contents

transreg-package	2
.residuals	3
.signdisc	4
calibrate	4
coef.transreg	5
compare	7
extract	8
fitted.transreg	9
plot.transreg	10
predict.transreg	12
print.transreg	13
simulate	14
transreg	15
weights.transreg	17

<b>Index</b>	<b>19</b>
--------------	-----------

---

transreg-package	<i>Penalised regression with multiple sets of prior effects</i>
------------------	---

---

### Description

The R package ‘transreg’ implements penalised regression with multiple sets of prior effects.

### Details

Use function [transreg()] for model fitting. Type ‘library(transreg)’ and then ‘?transreg’ or ‘help("transreg")’ to open its help file.

See the vignette for further examples. Type ‘vignette("transreg")’ or ‘browseVignettes("transreg")’ to open the vignette.

### Author(s)

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

### References

Armin Rauschenberger, Zied Landoulsi, Mark A. van de Wiel, and Enrico Glaab (2023). "Penalised regression with multiple sets of prior effects". *Bioinformatics* 39(12):btad680. doi:10.1093/bioinformatics/btad680. (Click [here](#) to access PDF.)

### See Also

Useful links:

- <https://github.com/rauschenberger/transreg>
- <https://rauschenberger.github.io/transreg/>
- Report bugs at <https://github.com/rauschenberger/transreg/issues>

### Examples

```
?transreg  
?predict.transreg  
?coef.transreg
```

---

.residuals                      *Calculate residuals*

---

### Description

Calculates residuals from observed outcome and predicted values (Gaussian family) or predicted probabilities (binomial family). Called by `‘.exp.multiple‘` and `‘.iso.multiple‘`.

### Usage

```
.residuals(y, y_hat, family)
```

### Arguments

<code>y</code>	response: vector of length $n$ (see family)
<code>y_hat</code>	predicted values or probabilities (see family): vector of length $n$ , or matrix with $n$ rows (samples) and $k$ columns (methods)
<code>family</code>	character "gaussian" ( $y$ : real numbers, $y\_hat$ : real numbers) or "binomial" ( $y$ : 0s and 1s, $y\_hat$ : unit interval)

### Examples

```
n <- 100  
p <- 5  
X <- matrix(stats::rnorm(n*p),nrow=n,ncol=p)  
#y <- stats::rbinom(n,size=1,prob=0.5)  
y <- stats::rnorm(n)  
glm <- glm(y~X,family="gaussian")  
res <- residuals.glm(glm)  
y_hat <- predict(glm,type="response")  
all.equal(res,y-y_hat)
```

---

`.signdisc`

*Sign discovery*

---

### Description

Assigns signs to prior weights to obtain prior coefficients

### Usage

```
.signdisc(y, X, prior, family, foldid = NULL, nfolds = 10, track = FALSE)
```

### Arguments

<code>y</code>	target: vector of length $n$ (see family)
<code>X</code>	features: matrix with $n$ rows (samples) and $p$ columns (features)
<code>prior</code>	prior coefficients: matrix with $p$ rows (features) and $k$ columns (sources of co-data)
<code>family</code>	character "gaussian" ( $y$ : real numbers), "binomial" ( $y$ : 0s and 1s), or "poisson" ( $y$ : non-negative integers);
<code>foldid</code>	fold identifiers: vector of length $n$ with entries from 1 to 'nfolds'
<code>nfolds</code>	number of folds: positive integer
<code>track</code>	show intermediate output (messages and plots): logical

---

`calibrate`

*Internal functions*

---

### Description

Internal functions called by [`transreg()`], depending on choice between exponential and isotonic calibration.

### Usage

```
.exp.multiple(
  y,
  X,
  prior,
  family,
  switch = FALSE,
  select = TRUE,
  track = FALSE
)
```

```

.iso.multiple(
  y,
  X,
  prior,
  family,
  switch = FALSE,
  select = TRUE,
  track = FALSE
)

.iso.fast.single(y, X, prior, family)

.iso.slow.single(y, X, prior, family)

```

### Arguments

y	target: vector of length $n$ (see family)
X	features: matrix with $n$ rows (samples) and $p$ columns (features)
prior	prior coefficients: matrix with $p$ rows (features) and $k$ columns (sources of co-data)
family	character "gaussian" ( $y$ : real numbers), "binomial" ( $y$ : 0s and 1s), or "poisson" ( $y$ : non-negative integers);
switch	choose between positive and negative weights for each source: logical
select	select from sources: logical
track	show intermediate output (messages and plots): logical

### Functions

- `.exp.multiple()`: called by 'transreg' if 'scale="exp"'
- `.iso.multiple()`: called by 'transreg' if 'scale="iso"'
- `.iso.fast.single()`: called by 'transreg' if 'scale="iso"' (via '`.iso.multiple()`')
- `.iso.slow.single()`: replaced by '`.iso.fast.single()`'

### See Also

Use `[transreg()]` for model fitting.

---

coef.transreg

*Extract Coefficients*

---

### Description

Extracts coefficients from an object of class `[transreg]`.

**Usage**

```
## S3 method for class 'transreg'
coef(object, stack = NULL, ...)
```

**Arguments**

object	object of class ‘transreg’
stack	character "sta" (standard stacking) or "sim" (simultaneous stacking)
...	(not applicable)

**Value**

Returns estimated coefficients. The output is a list with two slots: slot ‘alpha’ with the estimated intercept (scalar), and slot ‘beta’ with the estimated slopes (vector).

**References**

[Armin Rauschenberger, Zied Landoulsi, Mark A. van de Wiel, and Enrico Glaab \(2023\). "Penalised regression with multiple sets of prior effects". \*Bioinformatics\* 39\(12\):btad680. doi:10.1093/bioinformatics/btad680.](#) (Click [here](#) to access PDF.)

**See Also**

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

**Examples**

```
#--- simulation ---
set.seed(1)
n <- 100; p <- 500
X <- matrix(rnorm(n=n*p),nrow=n,ncol=p)
beta <- rnorm(p)
prior <- beta + rnorm(p)
y <- X %*% beta

#--- glmnet (without prior effects) ---
object <- glmnet::cv.glmnet(y=y,x=X,alpha=0)
beta_hat <- coef(object,s="lambda.min")[-1]
mean((beta-beta_hat)^2)

#--- transreg (with prior effects) ---
object <- transreg(y=y,X=X,prior=prior,alpha=0)
beta_hat <- coef(object)$beta
mean((beta-beta_hat)^2) # decrease in MSE?
```

---

compare	<i>Cross-validation (reproducibility)</i>
---------	---

---

### Description

Function for reproducing hold-out method (simulation) and  $k$ -fold cross-validation (application). See vignette.

### Usage

```
compare(
  target,
  source = NULL,
  prior = NULL,
  z = NULL,
  family,
  alpha,
  scale = "iso",
  sign = FALSE,
  switch = FALSE,
  select = TRUE,
  foldid.ext = NULL,
  nfolds.ext = 10,
  foldid.int = NULL,
  nfolds.int = 10,
  type.measure = "deviance",
  alpha.prior = NULL,
  naive = TRUE,
  seed = NULL,
  cores = 1,
  xrnet = FALSE
)
```

### Arguments

target	list with slot x (feature matrix with n rows and p columns) and slot y (target vector of length n)
source	list of k lists, each with slot x (feature matrix with m <sub>i</sub> rows and p columns) and slot y (target vector of length m <sub>i</sub> )
prior	prior coefficients: matrix with p rows (features) and k columns (sources of co-data)
z	prior weights
family	character "gaussian" (y: real numbers), "binomial" (y: 0s and 1s), or "poisson" (y: non-negative integers);
alpha	elastic net mixing parameter (0=ridge, 1=lasso): number between 0 and 1

scale	character "exp" for exponential calibration or "iso" for isotonic calibration
sign	sign discovery procedure: logical (experimental argument)
switch	choose between positive and negative weights for each source: logical
select	select from sources: logical
foldid.ext	external fold identifiers
nfolds.ext	number of external folds
foldid.int	internal fold identifiers
nfolds.int	number of internal folds
type.measure	character
alpha.prior	alpha for source regression
naive	compare with naive transfer learning: logical
seed	random seed
cores	number of cores for parallel computing (requires R package 'doMC')
xrnet	compare with xrnet: logical

**See Also**

[transreg()]

---

extract

*Internal functions*

---

**Description**

Internal functions called by [coef.transreg()], [predict.transreg()] and [weights.transreg()], depending on choice between standard stacking and simultaneous stacking.

**Usage**

```
.predict.sta(object, newx, ...)
.predict.sim(object, newx, ...)
.coef.sta(object, ...)
.coef.sim(object, ...)
.weights.sta(object, ...)
.weights.sim(object, ...)
.which.stack(object, stack)
```



**Arguments**

object	object of class 'transreg'
newx	features: matrix with $n$ rows (samples) and $p$ columns (variables)
...	(not applicable)
stack	character "sta" (standard stacking) or "sim" (simultaneous stacking)

**Functions**

- `.predict.sta()`: called by 'predict.transreg' if 'stack="sta"'
- `.predict.sim()`: called by 'predict.transreg' if 'stack="sim"'
- `.coef.sta()`: called by 'coef.transreg' if 'stack="sta"'
- `.coef.sim()`: called by 'coef.transreg' if 'stack="sim"'
- `.weights.sta()`: called by 'weights.transreg' if 'stack="sta"'
- `.weights.sim()`: called by 'weights.transreg' if 'stack="sim"'
- `.which.stack()`: called by 'coef.transreg', 'predict.transreg' and 'weights.transreg'

**See Also**

Use [coef](#), [predict](#) and [weights](#).

---

fitted.transreg	<i>Fitted values</i>
-----------------	----------------------

---

**Description**

Extracts fitted values

**Usage**

```
## S3 method for class 'transreg'
fitted(object, stack = NULL, ...)
```

**Arguments**

object	object of class 'transreg'
stack	character "sta" (standard stacking) or "sim" (simultaneous stacking)
...	(not applicable)

**Value**

Returns fitted values. The output is a numerical vector with one entry for sample.

## References

Armin Rauschenberger, Zied Landoulsi, Mark A. van de Wiel, and Enrico Glaab (2023). "Penalised regression with multiple sets of prior effects". *Bioinformatics* 39(12):btad680. doi:10.1093/bioinformatics/btad680. (Click [here](#) to access PDF.)

## See Also

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

## Examples

```
#--- simulation ---
set.seed(1)
n0 <- 100; n1 <- 10000; n <- n0 + n1; p <- 500
X <- matrix(rnorm(n=n*p),nrow=n,ncol=p)
beta <- rnorm(p)
prior <- beta + rnorm(p)
y <- X %*% beta

#--- train-test split ---
foldid <- rep(c(0,1),times=c(n0,n1))
y0 <- y[foldid==0]
X0 <- X[foldid==0,]
y1 <- y[foldid==1]
X1 <- X[foldid==1,]

object <- transreg(y=y0,X=X0,prior=prior)

#--- fitted values ---
y0_hat <- fitted(object)
mean((y0-y0_hat)^2)

#--- predicted values ---
y1_hat <- predict(object,newx=X1)
mean((y1-y1_hat)^2) # increase in MSE?
```

---

plot.transreg

*Plot transreg-object*

---

## Description

Plot transreg-object

## Usage

```
## S3 method for class 'transreg'
plot(x, stack = NULL, ...)
```

**Arguments**

x	object of type transreg
stack	character "sta" (standard stacking) or "sim" (simultaneous stacking)
...	(not applicable)

**Value**

Returns four plots.

\* top-left: Calibrated prior effects ( $y$ -axis) against original prior effects ( $x$ -axis). Each line is for one source of prior effects, with the colour given by [grDevices::palette()] (black: 1, red: 2, green: 3, blue: 4, ...).

\* top-right: Estimated coefficients with transfer learning ( $y$ -axis) against estimated coefficients without transfer learning ( $x$ -axis). Each point represents one feature.

\* bottom-left: Estimated weights for sources of prior effects (labels 1 to  $k$ ), and either estimated weights for 'lambda.min' and 'lambda.lse' models (standard stacking) or estimated weights for features (simultaneous stacking).

\* bottom-right: Absolute deviance residuals ( $y$ -axis) against fitted values ( $x$ -axis). Each point represents one sample.

**References**

Armin Rauschenberger, Zied Landoulsi, Mark A. van de Wiel, and Enrico Glaab (2023). "Penalised regression with multiple sets of prior effects". *Bioinformatics* 39(12):btad680. doi:10.1093/bioinformatics/btad680. (Click [here](#) to access PDF.)

**See Also**

Methods for objects of class [transreg] include `coef` and `predict`.

**Examples**

```
#--- simulation ---
set.seed(1)
n <- 100; p <- 500
X <- matrix(rnorm(n=n*p),nrow=n,ncol=p)
beta <- rnorm(p) #*rbinom(n=n,size=1,prob=0.2)
prior1 <- beta + rnorm(p)
prior2 <- beta + rnorm(p)
prior3 <- rnorm(p)
prior4 <- rnorm(p)
y <- X %*% beta

prior <- cbind(prior1,prior2,prior3,prior4)
object <- transreg(y=y,X=X,prior=prior,alpha=0,stack=c("sta","sim"))

plot(object,stack="sta")
```

---

predict.transreg      *Make Predictions*

---

### Description

Predicts outcome

### Usage

```
## S3 method for class 'transreg'  
predict(object, newx, stack = NULL, ...)
```

### Arguments

object	object of class 'transreg'
newx	features: matrix with $n$ rows (samples) and $p$ columns (variables)
stack	character "sta" (standard stacking) or "sim" (simultaneous stacking)
...	(not applicable)

### Value

Returns predicted values or predicted probabilities. The output is a column vector with one entry for each sample.

### References

Armin Rauschenberger, Zied Landoulsi, Mark A. van de Wiel, and Enrico Glaab (2023). "Penalised regression with multiple sets of prior effects". *Bioinformatics* 39(12):btad680. doi:10.1093/bioinformatics/btad680. (Click [here](#) to access PDF.)

### See Also

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

### Examples

```
#--- simulation ---  
set.seed(1)  
n0 <- 100; n1 <- 10000; n <- n0 + n1; p <- 500  
X <- matrix(rnorm(n=n*p),nrow=n,ncol=p)  
beta <- rnorm(p)  
prior <- beta + rnorm(p)  
y <- X %*% beta  
  
#--- train-test split ---  
foldid <- rep(c(0,1),times=c(n0,n1))  
y0 <- y[foldid==0]  
X0 <- X[foldid==0,]
```

```

y1 <- y[foldid==1]
X1 <- X[foldid==1,]

#--- glmnet (without prior effects) ---
object <- glmnet::cv.glmnet(y=y0,x=X0)
y_hat <- predict(object,newx=X1,s="lambda.min")
mean((y1-y_hat)^2)

#--- transreg (with prior effects) ---
object <- transreg(y=y0,X=X0,prior=prior)
y_hat <- predict(object,newx=X1)
mean((y1-y_hat)^2) # decrease in MSE?

```

---

```
print.transreg      Print transreg-object
```

---

## Description

Show summary of transreg-object

## Usage

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

## Arguments

x	object of class transreg
...	(not applicable)

## Value

Returns family of distributions, elastic net mixing parameter (*alpha*), number of samples (*n*), number of features (*p*), number of sources of co-data (*k*), chosen calibration method (exponential or isotonic), and chosen stacking method (standard or simultaneous).

## Examples

```

#--- simulation ---
set.seed(1)
n <- 100; p <- 500
X <- matrix(rnorm(n=n*p),nrow=n,ncol=p)
beta <- rnorm(p)
prior <- beta + rnorm(p)
y <- X %*% beta

#--- print.transreg ---
object <- transreg(y=y,X=X,prior=prior)
object

```

---

 simulate

*Simulation (reproducibility)*


---

**Description**

Function for reproducing 'internal' simulation study. See vignette.

**Usage**

```
simulate(
  p = 1000,
  n.target = 100,
  n.source = 150,
  k = 2,
  family = "gaussian",
  prop = 0.01,
  rho.beta = 0.95,
  rho.x = 0.95,
  w = 0.5,
  trans = rep(TRUE, times = k),
  exp = rep(1, times = k)
)
```

**Arguments**

p	number of features
n.target	sample size for target data set
n.source	sample size(s) for source data set(s), scalar or vector of length k
k	number of source data sets
family	"Gaussian", "binomial" or "poisson"
prop	approximate proportion of features with effects
rho.beta	correlation between effects (across different data sets)
rho.x	base for decreasing correlation structure for correlation between features
w	weight between signal and noise
trans	logical vector of length $k$ : transferable (TRUE) or non-transferable (FALSE) source
exp	non-negative vector of length $k$ for transforming beta to $\text{sign}(\text{beta}) * \text{abs}(\text{beta})^{\text{exp}}$

**See Also**

Use [glmtrans::models()] for reproducing 'external' simulation study.

---

transreg *Penalised regression with multiple sets of prior effects*

---

### Description

Implements penalised regression with multiple sets of prior effects

### Usage

```
transreg(
  y,
  X,
  prior,
  family = "gaussian",
  alpha = 1,
  foldid = NULL,
  nfolds = 10,
  scale = "iso",
  stack = "sim",
  sign = FALSE,
  switch = FALSE,
  select = TRUE,
  track = FALSE,
  parallel = FALSE
)
```

### Arguments

y	target: vector of length $n$ (see family)
X	features: matrix with $n$ rows (samples) and $p$ columns (features)
prior	prior coefficients: matrix with $p$ rows (features) and $k$ columns (sources of co-data)
family	character "gaussian" ( $y$ : real numbers), "binomial" ( $y$ : 0s and 1s), or "poisson" ( $y$ : non-negative integers);
alpha	elastic net mixing parameter (0=ridge, 1=lasso): number between 0 and 1
foldid	fold identifiers: vector of length $n$ with entries from 1 to 'nfolds'
nfolds	number of folds: positive integer
scale	character "exp" for exponential calibration or "iso" for isotonic calibration
stack	character "sta" (standard stacking) or "sim" (simultaneous stacking)
sign	sign discovery procedure: logical (experimental argument)
switch	choose between positive and negative weights for each source: logical
select	select from sources: logical
track	show intermediate output (messages and plots): logical
parallel	logical (see cv.glmnet)

**Details**

\*  $n$ : sample size \*  $p$ : number of features \*  $k$ : number of sources

**Value**

Returns an object of class 'transreg'. Rather than accessing its slots (see list below), it is recommended to use methods like [coef.transreg()] and [predict.transreg()].

\* slot 'base': Object of class 'glmnet'. Regression of outcome on features (without prior effects), with  $1 + p$  estimated coefficients (intercept + features).

\* slot 'meta.sta': 'NULL' or object of class 'glmnet'. Regression of outcome on cross-validated linear predictors from prior effects and estimated effects, with  $1 + k + 2$  estimated coefficients (intercept + sources of co-data + lambda\_min and lambda\_1se).

\* slot 'meta.sim': 'NULL' or object of class 'glmnet'. Regression of outcome on meta-features (cross-validated linear predictors from prior effects) and original features, with  $1 + k + p$  estimated coefficients (intercept + sources of co-data + features).

\* slot 'prior.calib': Calibrated prior effects. Matrix with  $p$  rows and  $k$  columns.

\* slot 'data': Original data. List with slots 'y', 'X' and 'prior' (see arguments).

\* slot 'info': Information on call. Data frame with entries  $n$ ,  $p$ ,  $k$ , 'family', 'alpha', 'scale' and 'stack' (see details and arguments).

**References**

Armin Rauschenberger, Zied Landoulsi, Mark A. van de Wiel, and Enrico Glaab (2023). "Penalised regression with multiple sets of prior effects". *Bioinformatics* 39(12):btad680. doi:10.1093/bioinformatics/btad680. (Click [here](#) to access PDF.)

**See Also**

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

**Examples**

```
#--- simulation ---
n <- 100; p <- 500
X <- matrix(rnorm(n=n*p),nrow=n,ncol=p)
beta <- rnorm(p)*rbinom(n=p,size=1,prob=0.2)
prior1 <- beta + rnorm(p)
prior2 <- beta + rnorm(p)
y_lin <- X %*% beta
y_log <- 1*(y_lin > 0)

#--- single vs multiple priors ---
one <- transreg(y=y_lin,X=X,prior=prior1)
two <- transreg(y=y_lin,X=X,prior=cbind(prior1,prior2))
weights(one)
weights(two)
```



```

#--- linear vs logistic regression ---
lin <- transreg(y=y_lin,X=X,prior=prior1,family="gaussian")
log <- transreg(y=y_log,X=X,prior=prior1,family="binomial")
hist(predict(lin,newx=X)) # predicted values
hist(predict(log,newx=X)) # predicted probabilities

#--- ridge vs lasso penalisation ---
ridge <- transreg(y=y_lin,X=X,prior=prior1,alpha=0)
lasso <- transreg(y=y_lin,X=X,prior=prior1,alpha=1)
# initial coefficients (without prior)
plot(x=coef(ridge$base)[-1]) # dense
plot(x=coef(lasso$base)[-1]) # sparse
# final coefficients (with prior)
plot(x=coef(ridge)$beta) # dense
plot(x=coef(lasso)$beta) # not sparse

#--- exponential vs isotonic calibration ---
exp <- transreg(y=y_lin,X=X,prior=prior1,scale="exp")
iso <- transreg(y=y_lin,X=X,prior=prior1,scale="iso")
plot(x=prior1,y=exp$prior.calib)
plot(x=prior1,y=iso$prior.calib)

#--- standard vs simultaneous stacking ---
prior <- c(prior1[1:250],rep(0,250))
sta <- transreg(y=y_lin,X=X,prior=prior,stack="sta")
sim <- transreg(y=y_lin,X=X,prior=prior,stack="sim")
plot(x=coef(sta$base)[-1],y=coef(sta)$beta)
plot(x=coef(sim$base)[-1],y=coef(sim)$beta)

```

---

weights.transreg	<i>Extract Weights</i>
------------------	------------------------

---

## Description

Extracts weights from an object of class [transreg].

## Usage

```
## S3 method for class 'transreg'
weights(object, stack = NULL, ...)
```

## Arguments

object	object of class 'transreg'
stack	character "sta" (standard stacking) or "sim" (simultaneous stacking)
...	(not applicable)

**Value**

Returns weights. The output is a numerical vector with one entry for each source of co-data.

**References**

Armin Rauschenberger, Zied Landoulsi, Mark A. van de Wiel, and Enrico Glaab (2023). "Penalised regression with multiple sets of prior effects". *Bioinformatics* 39(12):btad680. doi:10.1093/bioinformatics/btad680. (Click [here](#) to access PDF.)

**See Also**

This function is about weights for sources of prior effects. To extract weights for features (estimated regression coefficients), use `[coef()]`.

**Examples**

```
#--- simulation ---
set.seed(1)
n <- 100; p <- 500
X <- matrix(rnorm(n=n*p),nrow=n,ncol=p)
beta <- rnorm(p)
prior <- cbind(beta+rnorm(p),beta+rnorm(p),rnorm(p),rnorm(p))
y <- X %*% beta

#--- example ---
object <- transreg(y=y,X=X,prior=prior)
weights(object)
```

# Index

## \* **documentation**

- transreg-package, 2
- .coef.sim (extract), 8
- .coef.sta (extract), 8
- .exp.multiple (calibrate), 4
- .iso.fast.single (calibrate), 4
- .iso.multiple (calibrate), 4
- .iso.slow.single (calibrate), 4
- .predict.sim (extract), 8
- .predict.sta (extract), 8
- .residuals, 3
- .signdisc, 4
- .weights.sim (extract), 8
- .weights.sta (extract), 8
- .which.stack (extract), 8

- calibrate, 4
- coef, 6, 9–12, 16
- coef.transreg, 5
- compare, 7

- extract, 8

- fitted.transreg, 9

- plot.transreg, 10
- predict, 6, 9–12, 16
- predict.transreg, 12
- print.transreg, 13

- simulate, 14

- transreg, 15
- transreg-package, 2

- weights, 9
- weights.transreg, 17