

# Package: glmtrans (via r-universe)

September 10, 2024

**Type** Package

**Title** Transfer Learning under Regularized Generalized Linear Models

**Version** 2.0.0

**Description** We provide an efficient implementation for two-step multi-source transfer learning algorithms in high-dimensional generalized linear models (GLMs). The elastic-net penalized GLM with three popular families, including linear, logistic and Poisson regression models, can be fitted. To avoid negative transfer, a transferable source detection algorithm is proposed. We also provides visualization for the transferable source detection results. The relevant paper is available on arXiv: <[arXiv:2105.14328](https://arxiv.org/abs/2105.14328)>.

**Imports** glmnet, ggplot2, foreach, doParallel, caret, assertthat, formatR, stats

**License** GPL-2

**Depends** R (>= 3.5.0)

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Ye Tian [aut, cre], Yang Feng [aut]

**Maintainer** Ye Tian <[ye.t@columbia.edu](mailto:ye.t@columbia.edu)>

**Date/Publication** 2022-02-08 07:30:02 UTC

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

**RemoteUrl** <https://github.com/cran/glmtrans>

**RemoteRef** HEAD

**RemoteSha** 27cfd76926c7ff481b64ce82ad56371e15673400

## Contents

glmtrans . . . . .	2
glmtrans_inf . . . . .	6
models . . . . .	8
plot.glmtrans . . . . .	10
predict.glmtrans . . . . .	11
print.glmtrans . . . . .	13
source_detection . . . . .	13

<b>Index</b>	<b>17</b>
--------------	-----------

---

glmtrans	<i>Fit a transfer learning generalized linear model (GLM) with elasticnet regularization.</i>
----------	---

---

## Description

Fit a transfer learning generalized linear model through elastic net regularization with target data set and multiple source data sets. It also implements a transferable source detection algorithm, which helps avoid negative transfer in practice. Currently can deal with Gaussian, logistic and Poisson models.

## Usage

```
glmtrans(
  target,
  source = NULL,
  family = c("gaussian", "binomial", "poisson"),
  transfer.source.id = "auto",
  alpha = 1,
  standardize = TRUE,
  intercept = TRUE,
  nfolds = 10,
  cores = 1,
  valid.proportion = NULL,
  valid.nfolds = 3,
  lambda = c(transfer = "lambda.1se", debias = "lambda.min", detection = "lambda.1se"),
  detection.info = TRUE,
  target.weights = NULL,
  source.weights = NULL,
  C0 = 2,
  ...
)
```

**Arguments**

target	target data. Should be a list with elements x and y, where x indicates a predictor matrix with each row/column as a(n) observation/variable, and y indicates the response vector.
source	source data. Should be a list with some sublists, where each of the sublist is a source data set, having elements x and y with the same meaning as in target data.
family	response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian". <ul style="list-style-type: none"> <li>• "gaussian": Gaussian distribution.</li> <li>• "binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.</li> <li>• "poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.</li> </ul>
transfer.source.id	transferable source indices. Can be either a subset of {1, ..., length(source)}, "all" or "auto". Default = "auto". <ul style="list-style-type: none"> <li>• a subset of {1, ..., length(source)}: only transfer sources with the specific indices.</li> <li>• "all": transfer all sources.</li> <li>• "auto": run transferable source detection algorithm to automatically detect which sources to transfer. For the algorithm, refer to the documentation of function source_detection.</li> </ul>
alpha	the elasticnet mixing parameter, with $0 \leq \alpha \leq 1$ . The penalty is defined as $(1 - \alpha)/2 \ \beta\ _2^2 + \alpha \ \beta\ _1$ <p>. alpha = 1 encodes the lasso penalty while alpha = 0 encodes the ridge penalty. Default = 1.</p>
standardize	the logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is TRUE.
intercept	the logical indicator of whether the intercept should be fitted or not. Default = TRUE.
nfolds	the number of folds. Used in the cross-validation for GLM elastic net fitting procedure. Default = 10. Smallest value allowable is nfolds = 3.
cores	the number of cores used for parallel computing. Default = 1.
valid.proportion	the proportion of target data to be used as validation data when detecting transferable sources. Useful only when transfer.source.id = "auto". Default = NULL, meaning that the cross-validation will be applied.
valid.nfolds	the number of folds used in cross-validation procedure when detecting transferable sources. Useful only when transfer.source.id = "auto" and valid.proportion = NULL. Default = 3.

<code>lambda</code>	<p>a vector indicating the choice of lambdas in transferring, debiasing and detection steps. Should be a vector with names "transfer", "debias", and "detection", each component of which can be either "lambda.min" or "lambda.1se". Component transfer is the lambda (the penalty parameter) used in transferring step. Component debias is the lambda used in debiasing step. Component detection is the lambda used in the transferable source detection algorithm. Default choice of <code>lambda.transfer</code> and <code>lambda.detection</code> are "lambda.1se", while default <code>lambda.debias</code> = "lambda.min". If the user wants to change the default setting, input a vector with corresponding <code>lambda.transfer/lambda.debias/lambda.detection</code> names and corresponding values. Examples: <code>lambda = list(transfer = "lambda.min", debias = "lambda.1se")</code>; <code>lambda = list(transfer = "lambda.min", detection = "lambda.min")</code>.</p> <ul style="list-style-type: none"> <li>• "lambda.min": value of lambda that gives minimum mean cross-validated error in the sequence of lambda.</li> <li>• "lambda.1se": largest value of lambda such that error is within 1 standard error of the minimum.</li> </ul>
<code>detection.info</code>	the logistic flag indicating whether to print detection information or not. Useful only when <code>transfer.source.id = "auto"</code> . Default = TRUE.
<code>target.weights</code>	weight vector for each target instance. Should be a vector with the same length of target response. Default = NULL, which makes all instances equal-weighted.
<code>source.weights</code>	a list of weight vectors for the instances from each source. Should be a list with the same length of the number of sources. Default = NULL, which makes all instances equal-weighted.
$C_0$	the constant used in the transferable source detection algorithm. See Algorithm 2 in Tian, Y. and Feng, Y., 2021. Default = 2.
<code>...</code>	additional arguments.

## Value

An object with S3 class "glmtrans".

<code>beta</code>	the estimated coefficient vector.
<code>family</code>	the response type.
<code>transfer.source.id</code>	the transferable source index. If in the input, <code>transfer.source.id = 1:length(source)</code> or <code>transfer.source.id = "all"</code> , then the outputed <code>transfer.source.id = 1:length(source)</code> . If the inputed <code>transfer.source.id = "auto"</code> , only transferable source detected by the algorithm will be outputed.
<code>fitting.list</code>	<p>a list of other parameters of the fitted model.</p> <ul style="list-style-type: none"> <li>• <code>w_ath</code> estimator obtained from the transferring step.</li> <li>• <code>delta_ath</code> estimator obtained from the debiasing step.</li> <li>• <code>target.valid.loss</code> the validation (or cross-validation) loss on target data. Only available when <code>transfer.source.id = "auto"</code>.</li> <li>• <code>source.loss</code> the loss on each source data. Only available when <code>transfer.source.id = "auto"</code>.</li> <li>• <code>threshold</code> the threshold to determine transferability. Only available when <code>transfer.source.id = "auto"</code>.</li> </ul>

## References

- Tian, Y. and Feng, Y., 2021. *Transfer Learning under High-dimensional Generalized Linear Models*. *arXiv preprint arXiv:2105.14328*.
- Li, S., Cai, T.T. and Li, H., 2020. *Transfer learning for high-dimensional linear regression: Prediction, estimation, and minimax optimality*. *arXiv preprint arXiv:2006.10593*.
- Friedman, J., Hastie, T. and Tibshirani, R., 2010. *Regularization paths for generalized linear models via coordinate descent*. *Journal of statistical software*, 33(1), p.1.
- Zou, H. and Hastie, T., 2005. *Regularization and variable selection via the elastic net*. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2), pp.301-320.
- Tibshirani, R., 1996. *Regression shrinkage and selection via the lasso*. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), pp.267-288.

## See Also

[predict.glmtrans](#), [source\\_detection](#), [models](#), [plot.glmtrans](#), [cv.glmnet](#), [glmnet](#).

## Examples

```
set.seed(0, kind = "L'Ecuyer-CMRG")

# fit a linear regression model
D.training <- models("gaussian", type = "all", n.target = 100, K = 2, p = 500)
D.test <- models("gaussian", type = "target", n.target = 100, p = 500)
fit.gaussian <- glmtrans(D.training$target, D.training$source)
y.pred.glmtrans <- predict(fit.gaussian, D.test$target$x)

# compare the test MSE with classical Lasso fitted on target data
library(glmnet)
fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y)
y.pred.lasso <- predict(fit.lasso, D.test$target$x)

mean((y.pred.glmtrans - D.test$target$y)^2)
mean((y.pred.lasso - D.test$target$y)^2)

# fit a logistic regression model
D.training <- models("binomial", type = "all", n.target = 100, K = 2, p = 500)
D.test <- models("binomial", type = "target", n.target = 100, p = 500)
fit.binomial <- glmtrans(D.training$target, D.training$source, family = "binomial")
y.pred.glmtrans <- predict(fit.binomial, D.test$target$x, type = "class")

# compare the test error with classical Lasso fitted on target data
library(glmnet)
fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y, family = "binomial")
y.pred.lasso <- as.numeric(predict(fit.lasso, D.test$target$x, type = "class"))

mean(y.pred.glmtrans != D.test$target$y)
mean(y.pred.lasso != D.test$target$y)
```

```

# fit a Poisson regression model
D.training <- models("poisson", type = "all", n.target = 100, K = 2, p = 500)
D.test <- models("poisson", type = "target", n.target = 100, p = 500)
fit.poisson <- glmtrans(D.training$target, D.training$source, family = "poisson")
y.pred.glmtrans <- predict(fit.poisson, D.test$target$x, type = "response")

# compare the test MSE with classical Lasso fitted on target data
fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y, family = "poisson")
y.pred.lasso <- as.numeric(predict(fit.lasso, D.test$target$x, type = "response"))

mean((y.pred.glmtrans - D.test$target$y)^2)
mean((y.pred.lasso - D.test$target$y)^2)

```

---

glmtrans_inf	<i>Calculate asymptotic confidence intervals based on desparsified Lasso and two-step transfer learning method.</i>
--------------	---

---

### Description

Given the point estimate of the coefficient vector from `glmtrans`, calculate the asymptotic confidence interval of each component. The detailed inference algorithm can be found as Algorithm 3 in the latest version of Tian, Y. and Feng, Y., 2021. The algorithm is constructed based on a modified version of desparsified Lasso (Van de Geer, S. et al, 2014; Dezeure, R. et al, 2015).

### Usage

```

glmtrans_inf(
  target,
  source = NULL,
  family = c("gaussian", "binomial", "poisson"),
  beta.hat = NULL,
  nodewise.transfer.source.id = "all",
  cores = 1,
  level = 0.95,
  intercept = TRUE,
  ...
)

```

### Arguments

target	target data. Should be a list with elements <code>x</code> and <code>y</code> , where <code>x</code> indicates a predictor matrix with each row/column as a(n) observation/variable, and <code>y</code> indicates the response vector.
source	source data. Should be a list with some sublists, where each of the sublist is a source data set, having elements <code>x</code> and <code>y</code> with the same meaning as in target data.

family	response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian". <ul style="list-style-type: none"> <li>• "gaussian": Gaussian distribution.</li> <li>• "binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.</li> <li>• "poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.</li> </ul>
beta.hat	initial estimate of the coefficient vector (the intercept should be the first component). Can be from the output of function glmtrans.
nodewise.transfer.source.id	transferable source indices in the inference (the set A in Algorithm 3 of Tian, Y. and Feng, Y., 2021). Can be either a subset of {1, ..., length(source)}, "all" or NULL. Default = "all". <ul style="list-style-type: none"> <li>• a subset of {1, ..., length(source)}: only transfer sources with the specific indices.</li> <li>• "all": transfer all sources.</li> <li>• NULL: don't transfer any sources and only use target data.</li> </ul>
cores	the number of cores used for parallel computing. Default = 1.
level	the level of confidence interval. Default = 0.95. Note that the level here refers to the asymptotic level of confidence interval of a single component rather than the multiple intervals.
intercept	whether the model includes the intercept or not. Default = TRUE. Should be set as TRUE if the intercept of beta.hat is not zero.
...	additional arguments.

### Value

a list of output.	b.hat = b.hat, beta.hat = beta.hat, CI = CI, var.est = var.est
b.hat	the center of confidence intervals. A p-dimensional vector, where p is the number of predictors.
beta.hat	the initial estimate of the coefficient vector (the same as input).
CI	confidence intervals (CIs) with the specific level. A p by 3 matrix, where three columns indicate the center, lower limit and upper limit of CIs, respectively. Each row represents a coefficient component.
var.est	the estimate of variances in the CLT (Theta transpose times Sigma times Theta, in section 2.5 of Tian, Y. and Feng, Y., 2021). A p-dimensional vector, where p is the number of predictors.

### References

- Tian, Y. and Feng, Y., 2021. *Transfer Learning under High-dimensional Generalized Linear Models*. *arXiv preprint arXiv:2105.14328*.
- Van de Geer, S., Bühlmann, P., Ritov, Y.A. and Dezeure, R., 2014. *On asymptotically optimal confidence regions and tests for high-dimensional models*. *The Annals of Statistics*, 42(3), pp.1166-1202.

Dezeure, R., Bühlmann, P., Meier, L. and Meinshausen, N., 2015. *High-dimensional inference: confidence intervals, p-values and R-software hdi*. *Statistical science*, pp.533-558.

### See Also

[glmtrans](#).

### Examples

```
## Not run:
set.seed(0, kind = "L'Ecuyer-CMRG")

# generate binomial data
D.training <- models("binomial", type = "all", K = 2, p = 200)

# fit a logistic regression model via two-step transfer learning method
fit.binomial <- glmtrans(D.training$target, D.training$source, family = "binomial")

# calculate the CI based on the point estimate from two-step transfer learning method
fit.inf <- glmtrans_inf(target = D.training$target, source = D.training$source,
family = "binomial", beta.hat = fit.binomial$beta, cores = 2)

## End(Not run)
```

---

models

*Generate data from Gaussian, logistic and Poisson models.*

---

### Description

Generate data from Gaussian, logistic and Poisson models used in the simulation part of Tian, Y. and Feng, Y., 2021.

### Usage

```
models(
  family = c("gaussian", "binomial", "poisson"),
  type = c("all", "source", "target"),
  cov.type = 1,
  h = 5,
  K = 5,
  n.target = 200,
  n.source = rep(100, K),
  s = 5,
  p = 500,
  Ka = K
)
```



**Arguments**

family	response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian". <ul style="list-style-type: none"> <li>"gaussian": Gaussian distribution.</li> <li>"binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.</li> <li>"poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.</li> </ul>
type	the type of generated data. Can be "all", "source" or "target".
cov.type	the type of covariates. Can be 1 or 2 (numerical). If it equals to 1, the predictors will be generated from the distribution used in Section 4.1.1 (Ah-Trans-GLM) in the latest version of Tian, Y. and Feng, Y., 2021. If it equals to 2, the predictors will be generated from the distribution used in Section 4.1.2 (When transferable sources are unknown). <ul style="list-style-type: none"> <li>"all": generate a list with a target data set of size n. target and K source data set of size n. source.</li> <li>"source": generate a list with K source data set of size n. source.</li> <li>"target": generate a list with a target data set of size n. target.</li> </ul>
h	measures the deviation ( $l_1$ -norm) of transferable source coefficient from the target coefficient.
K	the number of source data sets. Default = 5.
n.target	the sample size of target data. Should be a positive integer. Default = 100.
n.source	the sample size of each source data. Should be a vector of length K. Default is a K-vector with all elements 150.
s	how many components in the target coefficient are non-zero, which controls the sparsity of target problem. Default = 15.
p	the dimension of data. Default = 1000.
Ka	the number of transferable sources. Should be an integer between 0 and K. Default = K.

**Value**

a list of data sets which depend on the value of type.

- type = "all": a list of two components named "target" and "source" storing the target and source data, respectively. Component source is a list containing K components with the first Ka ones h-transferable and the remaining ones h-nontransferable. The target data set and each source data set have components "x" and "y", as the predictors and responses, respectively.
- type = "source": a list with a single component "source". This component contains a list of K components with the first Ka ones h-transferable and the remaining ones h-nontransferable. Each source data set has components "x" and "y", as the predictors and responses, respectively.
- type = "target": a list with a single component "target". This component contains another list with components "x" and "y", as the predictors and responses of target data, respectively.

**References**

Tian, Y. and Feng, Y., 2021. *Transfer Learning under High-dimensional Generalized Linear Models*. *arXiv preprint arXiv:2105.14328*.

**See Also**

[glmtrans](#).

**Examples**

```
set.seed(0, kind = "L'Ecuyer-CMRG")

D.all <- models("binomial", type = "all")
D.target <- models("binomial", type = "target")
D.source <- models("binomial", type = "source")
```

---

plot.glmtrans	<i>Visualize the losses of different sources and the threshold to determine transferability.</i>
---------------	--

---

**Description**

Plot the losses of different sources and the threshold to determine transferability for object with class "glmtrans" or "glmtrans\_source\_detection".

**Usage**

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

**Arguments**

x            an object from class "glmtrans" or "glmtrans\_source\_detection", which are the output of functions `glmtrans` and `source_detection`, respectively.

...           additional arguments that can be passed to `ggplot` function.

**Value**

a "ggplot" visualization with the transferable threshold and losses of different sources.

**References**

Tian, Y. and Feng, Y., 2021. *Transfer learning with high-dimensional generalized linear models*. *Submitted*.

**See Also**

[glmtrans](#), [source\\_detection](#), [ggplot](#).

**Examples**

```
set.seed(1, kind = "L'Ecuyer-CMRG")

D.training <- models("gaussian", K = 2, p = 500, Ka = 1)

# plot for class "glmtrans"
fit.gaussian <- glmtrans(D.training$target, D.training$source)
plot(fit.gaussian)

# plot for class "glmtrans_source_detection"
detection.gaussian <- source_detection(D.training$target, D.training$source)
plot(detection.gaussian)
```

---

<code>predict.glmtrans</code>	<i>Predict for new data from a "glmtrans" object.</i>
-------------------------------	---

---

**Description**

Predict from a "glmtrans" object based on new observation data. There are various types of output available.

**Usage**

```
## S3 method for class 'glmtrans'
predict(
  object,
  newx,
  type = c("link", "response", "class", "integral response"),
  ...
)
```

**Arguments**

<code>object</code>	an object from class "glmtrans", which comes from the output of function <code>glmtrans</code> .
<code>newx</code>	the matrix of new values for predictors at which predictions are to be made. Should be in accordance with the data for training object.
<code>type</code>	the type of prediction. Default = "link".
<code>...</code>	additional arguments. <ul style="list-style-type: none"> <li>• "link" the linear predictors. When <code>family = "gaussian"</code>, it is the same as the predicted responses.</li> </ul>

- "response" gives the predicted probabilities when family = "binomial", the predicted mean when family = "poisson", and the predicted responses when family = "gaussian".
- "class" the predicted 0/1 responses for logistic distribution. Applies only when family = "binomial".
- "integral response" the predicted integral response for Poisson distribution. Applies only when family = "poisson".

### Value

the predicted result on new data, which depends on type.

### References

Tian, Y. and Feng, Y., 2021. *Transfer learning with high-dimensional generalized linear models*. Submitted.

### See Also

[glmtrans](#).

### Examples

```
set.seed(1, kind = "L'Ecuyer-CMRG")

# fit a logistic model
D.training <- models("binomial", type = "all", K = 1, p = 500)
D.test <- models("binomial", type = "target", n.target = 10, p = 500)
fit.binomial <- glmtrans(D.training$target, D.training$source, family = "binomial")

predict(fit.binomial, D.test$target$x, type = "link")
predict(fit.binomial, D.test$target$x, type = "response")
predict(fit.binomial, D.test$target$x, type = "class")

# fit a Poisson model
D.training <- models("poisson", type = "all", K = 1, p = 500)
D.test <- models("poisson", type = "target", n.target = 10, p = 500)
fit.poisson <- glmtrans(D.training$target, D.training$source, family = "poisson")

predict(fit.poisson, D.test$target$x, type = "response")
predict(fit.poisson, D.test$target$x, type = "integral response")
```

---

print.glmtrans      *Print a fitted "glmtrans" object.*

---

**Description**

Similar to the usual print methods, this function summarizes results from a fitted "glmtrans" object.

**Usage**

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

**Arguments**

x                    fitted "glmtrans" model object.  
...                  additional arguments.

**Value**

No value is returned.

**See Also**

[glmtrans](#).

**Examples**

```
set.seed(1, kind = "L'Ecuyer-CMRG")  
  
# fit a linear model  
D.training <- models("gaussian", K = 2, p = 500)  
fit.gaussian <- glmtrans(D.training$target, D.training$source)  
  
fit.gaussian
```

---

source\_detection      *Transferable source detection for GLM transfer learning algorithm.*

---

**Description**

Detect transferable sources from multiple source data sets. Currently can deal with Gaussian, logistic and Poisson models.

**Usage**

```

source_detection(
  target,
  source = NULL,
  family = c("gaussian", "binomial", "poisson"),
  alpha = 1,
  standardize = TRUE,
  intercept = TRUE,
  nfolds = 10,
  cores = 1,
  valid.nfolds = 3,
  lambda = "lambda.1se",
  detection.info = TRUE,
  target.weights = NULL,
  source.weights = NULL,
  C0 = 2,
  ...
)

```

**Arguments**

target	target data. Should be a list with elements x and y, where x indicates a predictor matrix with each row/column as a(n) observation/variable, and y indicates the response vector.
source	source data. Should be a list with some sublists, where each of the sublist is a source data set, having elements x and y with the same meaning as in target data.
family	response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian". <ul style="list-style-type: none"> <li>• "gaussian": Gaussian distribution.</li> <li>• "binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.</li> <li>• "poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.</li> </ul>
alpha	the elasticnet mixing parameter, with $0 \leq \alpha \leq 1$ . The penalty is defined as $(1 - \alpha)/2 \ \beta\ _2^2 + \alpha \ \beta\ _1$ <p>. alpha = 1 encodes the lasso penalty while alpha = 0 encodes the ridge penalty. Default = 1.</p>
standardize	the logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is TRUE.
intercept	the logical indicator of whether the intercept should be fitted or not. Default = TRUE.
nfolds	the number of folds. Used in the cross-validation for GLM elastic net fitting procedure. Default = 10. Smallest value allowable is nfolds = 3.

cores	the number of cores used for parallel computing. Default = 1.
valid.nfolds	the number of folds used in cross-validation procedure when detecting transferable sources. Useful only when <code>transfer.source.id = "auto"</code> . Default = 3.
lambda	lambda (the penalty parameter) used in the transferable source detection algorithm. Can be either "lambda.min" or "lambda.1se". Default = "lambda.1se".
detection.info	the logistic flag indicating whether to print detection information or not. Useful only when <code>transfer.source.id = "auto"</code> . Default = TRUE.
target.weights	weight vector for each target instance. Should be a vector with the same length of target response. Default = NULL, which makes all instances equal-weighted.
source.weights	a list of weight vectors for the instances from each source. Should be a list with the same length of the number of sources. Default = NULL, which makes all instances equal-weighted.
C0	the constant used in the transferable source detection algorithm. See Algorithm 2 in Tian, Y. and Feng, Y., 2021. Default = 2. <ul style="list-style-type: none"> <li>• "lambda.min": value of lambda that gives minimum mean cross-validated error in the sequence of lambda.</li> <li>• "lambda.1se": largest value of lambda such that error is within 1 standard error of the minimum.</li> </ul>
...	additional arguments.

### Value

An object with S3 class "glmtrans\_source\_detection".

target.valid.loss	the validation (or cross-validation) loss on target data. Only available when <code>transfer.source.id = "auto"</code> .
source.loss	the loss on each source data. Only available when <code>transfer.source.id = "auto"</code> .
threshold	the threshold to determine transferability. Only available when <code>transfer.source.id = "auto"</code> .

### Note

`source.loss` and `threshold` outputted by `source_detection` can be visualized by function `plot.glmtrans`.

### References

- Tian, Y. and Feng, Y., 2021. *Transfer Learning under High-dimensional Generalized Linear Models*. *arXiv preprint arXiv:2105.14328*.
- Li, S., Cai, T.T. and Li, H., 2020. *Transfer learning for high-dimensional linear regression: Prediction, estimation, and minimax optimality*. *arXiv preprint arXiv:2006.10593*.
- Friedman, J., Hastie, T. and Tibshirani, R., 2010. *Regularization paths for generalized linear models via coordinate descent*. *Journal of statistical software*, 33(1), p.1.

Zou, H. and Hastie, T., 2005. *Regularization and variable selection via the elastic net*. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2), pp.301-320.

Tibshirani, R., 1996. *Regression shrinkage and selection via the lasso*. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), pp.267-288.

### See Also

[glmtrans](#), [predict.glmtrans](#), [models](#), [plot.glmtrans](#), [cv.glmnet](#), [glmnet](#).

### Examples

```
set.seed(0, kind = "L'Ecuyer-CMRG")

# study the linear model
D.training <- models("gaussian", type = "all", K = 2, p = 500, Ka = 1, n.target = 100, cov.type = 2)
detection.gaussian <- source_detection(D.training$target, D.training$source)
detection.gaussian$transferable.source.id

# study the logistic model
D.training <- models("binomial", type = "all", K = 2, p = 500, Ka = 1, n.target = 100, cov.type = 2)
detection.binomial <- source_detection(D.training$target, D.training$source,
family = "binomial", cores = 2)
detection.binomial$transferable.source.id

# study Poisson model
D.training <- models("poisson", type = "all", K = 2, p = 500, Ka = 1, n.target = 100, cov.type = 2)
detection.poisson <- source_detection(D.training$target, D.training$source,
family = "poisson", cores = 2)
detection.poisson$transferable.source.id
```



# Index

`cv.glmnet`, [5](#), [16](#)

`ggplot`, [11](#)

`glmnet`, [5](#), [16](#)

`glmtrans`, [2](#), [8](#), [10–13](#), [16](#)

`glmtrans_inf`, [6](#)

`models`, [5](#), [8](#), [16](#)

`plot.glmtrans`, [5](#), [10](#), [16](#)

`plot.glmtrans_source_detection`  
(`plot.glmtrans`), [10](#)

`predict.glmtrans`, [5](#), [11](#), [16](#)

`print.glmtrans`, [13](#)

`source_detection`, [5](#), [11](#), [13](#)