

Package: tramnet (via r-universe)

October 17, 2026

Title Penalized Transformation Models

Version 0.1-0

URL <https://codeberg.org/LucasKook/tramnet>

BugReports <https://codeberg.org/LucasKook/tramnet/issues>

Description Partially penalized versions of specific transformation models implemented in package 'mlt'. Available models include a fully parametric version of the Cox model, other parametric survival models (Weibull, etc.), normal and transformed-normal (Box-Cox type) linear models, and continuous outcome logistic regression. Models can be specified via a formula interface. Hyperparameter tuning is facilitated through model-based optimization functionalities from package 'mlr3mbo'. The accompanying vignette describes the methodology used in 'tramnet' in detail. Transformation models and model-based optimization are described in Hothorn et al. (2019) <[doi:10.1111/sjos.12291](https://doi.org/10.1111/sjos.12291)> and Bischl et al. (2017) <[doi:10.48550/arXiv.1703.03373](https://doi.org/10.48550/arXiv.1703.03373)>, respectively.

Depends R (>= 4.2.0), tram (>= 1.4), CVXR (>= 1.8)

Imports mlt, basefun, sandwich, lhs, stats, graphics, grDevices, mlr3mbo (>= 1.2.0), mlr3, mlr3learners, bbotk (>= 1.0.0), paradox (>= 1.0.0), lgr

Suggests penalized, TH.data, survival, knitr, colorspace, mvtnorm, glmnet, trtf, Matrix, lattice, kableExtra, coin, tbm, DiceKriging, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

License GPL-2

NeedsCompilation no

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Config/pak/sysreqs cmake libgmp3-dev make pkg-config libclang-dev

Repository <https://lucaskook.r-universe.dev>
Date/Publication 2026-10-16 13:43:23 UTC
RemoteUrl <https://codeberg.org/lucaskook/tramnet>
RemoteRef HEAD
RemoteSha 2521887fb191ebbbdcd075736b076c5fdb83cbb7

Contents

coef.tramnet	2
coef.tramnet_Lm	3
Coxphnet	4
cvl_tramnet	7
elnet_obj	8
estfun.tramnet	9
lasso_obj	10
logLik.tramnet	11
mbo_recommended	12
mbo_tramnet	12
model.matrix.tramnet	14
plot.tramnet	15
plot_path	16
predict.tramnet	17
print.summary.tramnet	17
print.tramnet	18
prof_alpha	18
prof_lambda	19
residuals.tramnet	20
ridge_obj	21
simulate.tramnet	22
summary.tramnet	22
tramnet	23
Index	26

coef.tramnet	<i>coef method for class "tramnet"</i>
--------------	--

Description

coef method for class "tramnet"

Usage

```
## S3 method for class 'tramnet'  
coef(object, with_baseline = FALSE, tol = 1e-06, scaled = FALSE, ...)
```

Arguments

object	object of class "tramnet"
with_baseline	If TRUE, also prints coefficients for the baseline transformation
tol	tolerance when an estimate should be considered 0 and not returned (default: 1e-6)
scaled	only for models fitted via the formula interface: If FALSE (default), shift parameters are returned on the original scale of the covariates, otherwise on the scale of the internally scaled covariates (on which the penalty acts)
...	Additional arguments to coef

Value

Numeric vector containing the model shift parameter estimates

Author(s)

Torsten Hothorn, Lucas Kook

coef.tramnet_Lm	<i>coef method for class "tramnet_Lm"</i>
-----------------	---

Description

coef method for class "tramnet_Lm"

Usage

```
## S3 method for class 'tramnet_Lm'
coef(
  object,
  with_baseline = FALSE,
  tol = 1e-06,
  as.lm = FALSE,
  scaled = FALSE,
  ...
)
```

Arguments

object	object of class "tramnet_Lm"
with_baseline	If TRUE, also prints coefficients for the baseline transformation
tol	tolerance when an estimate should be considered 0 and not returned (default: 1e-6)
as.lm	If TRUE parameters are rescaled to the usual parametrization of lm

scaled	only for models fitted via the formula interface: If FALSE (default), shift parameters are returned on the original scale of the covariates, otherwise on the scale of the internally scaled covariates (on which the penalty acts)
...	Additional arguments to coef

Value

Numeric vector containing the linear model shift parameter estimates

Author(s)

Torsten Hothorn, Lucas Kook

Examples

```
data(cars)
m0 <- Lm(dist ~ 1, data = cars)
x <- as.matrix(cars[, "speed", drop = FALSE])
mt <- tramnet(m0, x = x, alpha = 0, lambda = 0)
coef(mt)
coef(mt, with_baseline = TRUE)
coef(mt, as.lm = TRUE)
coef(lm(dist ~ speed, data = cars))
```

Coxphnet

Regularised Transformation Models via Formulae

Description

Penalized versions of the transformation models implemented in package tram, specified via a formula. These are convenience wrappers around [tramnet](#), analogous to the corresponding functions [Coxph](#), [Colr](#), [Lm](#), [BoxCox](#), [Lehmann](#) and [Survreg](#).

Usage

```
Coxphnet(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  lambda,
  alpha,
  constraints = NULL,
  ...
)
```

```
)  
  
Colrnet(  
  formula,  
  data,  
  subset,  
  weights,  
  offset,  
  na.action = na.omit,  
  lambda,  
  alpha,  
  constraints = NULL,  
  ...  
)  
  
Lmnet(  
  formula,  
  data,  
  subset,  
  weights,  
  offset,  
  na.action = na.omit,  
  lambda,  
  alpha,  
  constraints = NULL,  
  ...  
)  
  
BoxCoxnet(  
  formula,  
  data,  
  subset,  
  weights,  
  offset,  
  na.action = na.omit,  
  lambda,  
  alpha,  
  constraints = NULL,  
  ...  
)  
  
Lehmannnet(  
  formula,  
  data,  
  subset,  
  weights,  
  offset,  
  na.action = na.omit,
```

```

    lambda,
    alpha,
    constraints = NULL,
    ...
)

Survregnet(
  formula,
  data,
  subset,
  weights,
  offset,
  na.action = na.omit,
  lambda,
  alpha,
  constraints = NULL,
  ...
)

```

Arguments

<code>formula</code>	an object of class "formula", see tram
<code>data</code>	an optional data frame containing the variables in the model
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process
<code>weights</code>	an optional vector of case weights to be used in the fitting process
<code>offset</code>	an optional vector of offsets
<code>na.action</code>	a function which indicates what should happen when the data contain NAs
<code>lambda</code>	a positive penalty parameter for the whole penalty function
<code>alpha</code>	a mixing parameter (between zero and one) defining the fraction between absolute and quadratic penalty terms
<code>constraints</code>	an optional list containing a matrix of linear inequality constraints on the regression coefficients (on the original scale of the covariates) and a vector specifying the rhs of the inequality
<code>...</code>	additional arguments to the corresponding function in package <code>tram</code> , e.g., <code>order</code> , <code>log_first</code> or <code>support</code>

Details

The left-hand side of `formula` (including `strata`, as in `y | s ~ x`) defines the unpenalized baseline transformation, all terms on the right-hand side are penalized. The design matrix is obtained via [model.matrix](#) without intercept and its columns are scaled to unit variance. The scaling of the training data is stored and re-used whenever newdata is supplied to one of the methods, or in [cvl_tramnet](#). Coefficients are reported on the original scale of the covariates, see [coef.tramnet](#).

Not all combinations of error distribution and censoring type lead to objective functions that can be solved by **CVXR**; see the table in the `tramnet` vignette.

Value

An object of class "tramnet", see [tramnet](#)

Author(s)

Lucas Kook

Examples

```
if (require("survival") & require("TH.data")) {
  data("GBSG2", package = "TH.data")
  GBSG2$surv <- with(GBSG2, Surv(time, cens))
  fit <- Coxphnet(surv ~ horTh + age + tsize + pnodes, data = GBSG2,
                 lambda = 5, alpha = 1, log_first = TRUE)

  coef(fit)
  coef(fit, scaled = TRUE)
  predict(fit, newdata = GBSG2[1:5, ], type = "survivor", q = 1000)
}
data("cars")
fit <- Lmnet(dist ~ speed, data = cars, lambda = 1, alpha = 0)
coef(fit, as.lm = TRUE)
```

cvl_tramnet

Cross validation for "tramnet" models

Description

k-fold cross validation for "tramnet" objects over a grid of the tuning parameters based on out-of-sample log-likelihood. For models fitted via the formula interface, the covariates are scaled within each training fold and the test fold is scaled accordingly.

Usage

```
cvl_tramnet(
  object,
  fold = 2,
  lambda = 0,
  alpha = 0,
  folds = NULL,
  fit_opt = FALSE
)
```

Arguments

object	object of class "tramnet"
fold	number of folds for cross validation
lambda	values for lambda to iterate over

alpha	values for alpha to iterate over
folds	manually specify folds for comparison with other methods
fit_opt	If TRUE, returns the full model evaluated at optimal hyper parameters

Value

Returns out-of-sample logLik and coefficient estimates for corresponding folds and values of the hyperparameters as an object of class "cvl_tramnet"

Author(s)

Lucas Kook

Examples

```
set.seed(241068)
if (require("survival") & require("TH.data")) {
  data("GBSG2", package = "TH.data")
  X <- 1 * matrix(GBSG2$horTh == "yes", ncol = 1)
  colnames(X) <- "horThyes"
  GBSG2$surv <- with(GBSG2, Surv(time, cens))
  m <- Coxph(surv ~ 1, data = GBSG2, log_first = TRUE)
  mt <- tramnet(model = m, x = X, lambda = 0, alpha = 0)
  mc <- Coxph(surv ~ horTh, data = GBSG2)
  cvl_tramnet(mt, fold = 2, lambda = c(0, 1), alpha = c(0, 1))
  ## formula interface
  mf <- Coxphnet(surv ~ horTh + age, data = GBSG2, lambda = 0, alpha = 0,
                 log_first = TRUE)
  cvl_tramnet(mf, fold = 2, lambda = c(0, 1), alpha = c(0, 1))
}
```

elnet_obj

Elastic net objective function for model based optimization

Description

This function generates an objective function for model-based optimization based on the cross-validated log-likelihood of a tramnet model with an elastic net penalty. It is not intended to be called by the user directly, instead it will be given as an argument to [mbo_tramnet](#).

Usage

```
elnet_obj(
  object,
  minlambda = 0,
  maxlambda = 16,
  minalpha = 0,
```

```

    maxalpha = 1,
    folds,
    noisy = FALSE,
    fold
  )

```

Arguments

object	object of class tramnet
minlambda	minimum value for lambda (default: 0)
maxlambda	maximum value for lambda (default: 16)
minalpha	minimum value for alpha (default: 0)
maxalpha	maximum value for alpha (default: 1)
folds	self specified folds for cross validation (mainly for reproducibility and comparability purposes)
noisy	indicates whether folds for k-fold cross-validation should be random for each iteration, leading to a noisy objective function (default: FALSE)
fold	fold for cross validation

Value

Single objective function for model based optimization of class [ObjectiveRFun](#).

estfun.tramnet	<i>estfun method for class "tramnet"</i>
----------------	--

Description

estfun method for class "tramnet" which computes the score contributions w.r.t. each model parameter.

Usage

```

## S3 method for class 'tramnet'
estfun(
  x,
  parm = coef(x, with_baseline = TRUE, tol = 0, scaled = TRUE),
  w = NULL,
  newdata,
  ...
)

```

Arguments

x	object of class "tramnet"
parm	parameters for evaluating the score
w	weights
newdata	data on which to compute the score contributions
...	additional arguments to estfun

Value

Matrix of score contributions w.r.t. model parameters evaluated at parm

Author(s)

Lucas Kook

lasso_obj	<i>Lasso objective function for model based optimization</i>
-----------	--

Description

This function generates an objective function for model-based optimization based on the cross-validated log-likelihood of a tramnet model with a lasso penalty only. It is not intended to be called by the user directly, instead it will be given as an argument to [mbo_tramnet](#).

Usage

```
lasso_obj(object, minlambda = 0, maxlambda = 16, folds, noisy = FALSE, fold)
```

Arguments

object	object of class tramnet
minlambda	minimum value for lambda (default: 0)
maxlambda	maximum value for lambda (default: 16)
folds	self specified folds for cross validation (mainly for reproducibility and comparability purposes)
noisy	indicates whether folds for k-fold cross-validation should be random for each iteration, leading to a noisy objective function (default: FALSE)
fold	fold for cross validation

Value

Single objective function for model based optimization of class [ObjectiveRFun](#).

logLik.tramnet	<i>logLik method for class "tramnet"</i>
----------------	--

Description

logLik method for class "tramnet"

Usage

```
## S3 method for class 'tramnet'
logLik(
  object,
  parm = coef(object, tol = 0, with_baseline = TRUE, scaled = TRUE),
  w = NULL,
  newdata,
  ...
)
```

Arguments

object	object of class "tramnet"
parm	parameters to evaluate the log likelihood at (for models fitted via the formula interface, shift parameters refer to the scaled covariates)
w	weights, defaults to the weights used for fitting object if newdata is missing
newdata	data to evaluate the log likelihood at
...	Additional arguments to logLik

Value

returns potentially weighted (w) log-likelihood based on object evaluated at parameters parm and data newdata. If newdata is missing, the penalized log-likelihood of the training data is returned, otherwise the (unpenalized) out-of-sample log-likelihood

Author(s)

Lucas Kook, Torsten Hothorn

mbo_recommended	<i>Fit recommended regularized tram based on model based optimization output</i>
-----------------	--

Description

Extracts the "optimal" tuning parameters from an object returned by `mbo_tramnet` and fits the corresponding tramnet model

Usage

```
mbo_recommended(mbo_obj, m0, x, ...)
```

Arguments

mbo_obj	object return by mbo_tramnet
m0	null model of class "tram" or a fitted model of class "tramnet", which is refitted at the recommended tuning parameters (in this case, x and ... are ignored)
x	matrix of covariables
...	additional arguments to tramnet()

Value

Object of class "tramnet"

mbo_tramnet	<i>Model based optimization for regularized transformation models</i>
-------------	---

Description

Uses model based optimization to find the optimal tuning parameter(s) in a regularized transformation model based on cross-validated log-likelihoods. Here the tramnet package makes use of the mlr3mbo interface for Bayesian Optimization in machine learning problems to maximize the cv-logLik as a black-box function of the tuning parameters alpha and lambda.

Usage

```
mbo_tramnet(
  object,
  fold = 2,
  n_design = 5,
  n_iter = 5,
  minlambda = 0,
  maxlambda = 16,
  minalpha = 0,
```

```

    maxalpha = 1,
    folds = NULL,
    learner = "regr.km",
    pred.type = "se",
    opt_crit = "ei",
    noisy = FALSE,
    obj_type = c("lasso", "ridge", "elnet"),
    verbose = TRUE,
    ...
)

```

Arguments

object	object of class tramnet
fold	fold for cross validation
n_design	number of rows of the initial design matrix based on a random latin hypercube design
n_iter	number of iterations in the model based optimization procedure
minlambda	minimum value for lambda (default: 0)
maxlambda	maximum value for lambda (default: 16)
minalpha	minimum value for alpha (default: 0)
maxalpha	maximum value for alpha (default: 1)
folds	self specified folds for cross validation (mainly for reproducibility and comparability purposes)
learner	type of learner used as surrogate model in the optimization, either the key of a regression learner in mlr_learners (default: "regr.km", a Kriging model, which requires package DiceKriging) or an object of class LearnerRegr
pred.type	prediction type of the learner (default: "se")
opt_crit	optimization criterion, either the key of an acquisition function in mlr_acqfunctions or an object of class AcqFunction , default: expected improvement ("ei")
noisy	indicates whether folds for k-fold cross-validation should be random for each iteration, leading to a noisy objective function (default: FALSE)
obj_type	objective type, one of "lasso", "ridge" or "elnet"
verbose	toggle for a verbose output (default: TRUE)
...	additional arguments are ignored

Value

returns a list containing the recommended tuning parameters (x), the corresponding objective value (y), i.e., minus two times the cross-validated log-likelihood, the type of objective (obj_type) and the optimization instance (instance) of class [OptimInstanceBatchSingleCrit](#), which contains the full archive of evaluated tuning parameters

Author(s)

Lucas Kook

Examples

```

if (require("survival") & require("TH.data") & require("DiceKriging")) {
  set.seed(241068)
  data("GBSG2", package = "TH.data")
  GBSG2$surv <- with(GBSG2, Surv(time, cens))
  mt <- Coxphnet(surv ~ horTh + age + tsize + pnodes, data = GBSG2,
                 lambda = 0, alpha = 0, log_first = TRUE)
  tmbo <- mbo_tramnet(mt, obj_type = "lasso", fold = 2, n_iter = 3,
                     verbose = FALSE)

  tmbo$x
  coef(mbo_recommended(tmbo, mt))
}

```

model.matrix.tramnet *model.matrix method for class "tramnet"*

Description

model.matrix method for class "tramnet", returning the (scaled) design matrix of the penalized covariates

Usage

```

## S3 method for class 'tramnet'
model.matrix(object, newdata, ...)

```

Arguments

object	object of class "tramnet"
newdata	optional data frame. For models fitted via the formula interface, the design matrix is computed from newdata and scaled as the training data. Not available for models fitted via the default interface
...	additional arguments are ignored

Value

Numeric matrix of covariates as used for fitting object (default) or computed from newdata

Author(s)

Lucas Kook

plot.tramnet	<i>plot method for class "tramnet"</i>
--------------	--

Description

plot method for class "tramnet"

Usage

```
## S3 method for class 'tramnet'
plot(
  x,
  newdata,
  type = c("distribution", "survivor", "density", "logdensity", "hazard", "loghazard",
    "cumhazard", "quantile", "trafo"),
  q = NULL,
  prob = 1:(K - 1)/K,
  K = 50,
  col = rgb(0.1, 0.1, 0.1, 0.1),
  lty = 1,
  add = FALSE,
  ...
)
```

Arguments

x	object of class "tramnet"
newdata	data used to predict and plot
type	type of plot produced
q	vector of quantiles
prob	vector of probabilities
K	number of data points to plot
col	see plot
lty	see plot
add	see plot
...	additional options to plot

Value

None

Author(s)

Lucas Kook

plot_path

*Plot regularization paths for "prof_*" classes***Description**

Plot regularization paths and optionally log-likelihood trajectories of objects of class "prof_alpha" and "prof_lambda". Coefficient names are automatically added to the plot.

Usage

```
plot_path(object, plot_logLik = FALSE, ...)
```

Arguments

object	object of class "prof_alpha" or "prof_lambda"
plot_logLik	Whether logLik trajectory should be plotted (default: FALSE)
...	additional arguments to plot

Value

None

Author(s)

Lucas Kook

Examples

```
library("tramnet")
if (require("survival") & require("penalized")) {
  data("nki70", package = "penalized")
  nki70$resp <- with(nki70, Surv(time, event))
  x <- scale(model.matrix(~ 0 + DIAPH3 + NUSAP1 + TSPYL5 + C20orf46, data = nki70))
  y <- Coxph(resp ~ 1, data = nki70, order = 10, log_first = TRUE)
  fit1 <- tramnet(y, x, lambda = 0, alpha = 1)
  pfl <- prof_lambda(fit1)
  plot_path(pfl)
  fit2 <- tramnet(y, x, lambda = 1, alpha = 1)
  pfa <- prof_alpha(fit2)
  plot_path(pfa)
}
```

predict.tramnet	<i>predict method for class "tramnet"</i>
-----------------	---

Description

predict method for class "tramnet"

Usage

```
## S3 method for class 'tramnet'
predict(object, newdata, ...)
```

Arguments

object	object of class "tramnet"
newdata	data used for prediction, defaults to the data used for fitting object. For models fitted via the formula interface, newdata contains the original (unscaled) covariates
...	Additional arguments to predict.ctm

Value

Vector of predictions based on object evaluated at each row of newdata

Author(s)

Lucas Kook

print.summary.tramnet	<i>print summary method for class "tramnet"</i>
-----------------------	---

Description

print summary method for class "tramnet"

Usage

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

Arguments

x	object of class "tramnet"
digits	number of digits to print
...	additional arguments

Value

prints textual summary in the console and returns an invisible copy of the "tramnet" object

Author(s)

Lucas Kook

print.tramnet	<i>print method for class "tramnet"</i>
---------------	---

Description

print method for class "tramnet"

Usage

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

Arguments

x	object of class "tramnet"
...	additional arguments to summary

Value

prints textual summary in the console and returns an invisible copy of the "tramnet" object

Author(s)

Lucas Kook

prof_alpha	<i>Profiling tuning parameters</i>
------------	------------------------------------

Description

Computes the regularization path of all coefficients for a single tuning, alpha, parameter over a sequence of values.

Usage

```
prof_alpha(model, min_alpha = 0, max_alpha = 1, nprof = 5, as.lm = FALSE)
```

Arguments

model	model of class tramnet
min_alpha	minimal value of alpha (default = 0)
max_alpha	maximal value of alpha (default = 15)
nprof	number of profiling steps (default = 5)
as.lm	return scaled coefficients for class "tramnet_Lm"

Value

Object of class "prof_alpha" which contains the regularization path of all coefficients and the log-likelihood over the mixing parameter alpha

Author(s)

Lucas Kook

Examples

```
library("tramnet")

if (require("survival") & require("penalized")) {
  data("nki70", package = "penalized")
  nki70$resp <- with(nki70, Surv(time, event))
  x <- scale(model.matrix(~ 0 + DIAPH3 + NUSAP1 + TSPYL5 + C20orf46, data = nki70))
  y <- Coxph(resp ~ 1, data = nki70, order = 10, log_first = TRUE)
  fit <- tramnet(y, x, lambda = 1, alpha = 1)
  pfa <- prof_alpha(fit)
  plot_path(pfa)
}
```

prof_lambda	<i>Profiling tuning parameters</i>
-------------	------------------------------------

Description

Computes the regularization path of all coefficients for a single tuning parameter, lambda, over a sequence of values.

Usage

```
prof_lambda(model, min_lambda = 0, max_lambda = 15, nprof = 5, as.lm = FALSE)
```

Arguments

<code>model</code>	model of class "tramnet"
<code>min_lambda</code>	minimal value of lambda (default = 0)
<code>max_lambda</code>	maximal value of lambda (default = 15)
<code>nprof</code>	number of profiling steps (default = 5)
<code>as.lm</code>	return scaled coefficients for class "tramnet_Lm"

Value

Object of class "prof_lambda" which contains the regularization path of all coefficients and the log-likelihood over the penalty parameter lambda

Author(s)

Lucas Kook

Examples

```
library("tramnet")
if (require("survival") & require("penalized")) {
  data("nki70", package = "penalized")
  nki70$resp <- with(nki70, Surv(time, event))
  x <- scale(model.matrix(~ 0 + DIAPH3 + NUSAP1 + TSPYL5 + C20orf46, data = nki70))
  y <- Coxph(resp ~ 1, data = nki70, order = 10, log_first = TRUE)
  fit <- tramnet(y, x, lambda = 0, alpha = 1)
  pfl <- prof_lambda(fit)
  plot_path(pfl)
}
```

residuals.tramnet	<i>residuals method for class "tramnet"</i>
-------------------	---

Description

residuals method for class "tramnet"

Usage

```
## S3 method for class 'tramnet'
residuals(
  object,
  parm = coef(object, tol = 0, with_baseline = TRUE, scaled = TRUE),
  w = NULL,
  newdata,
  ...
)
```

Arguments

object	object of class "tramnet"
parm	parameters to evaluate score at
w	weights
newdata	data to evaluate score at
...	additional arguments to residuals

Value

Returns a numeric vector of residuals for each row in newdata

Author(s)

Lucas Kook

ridge_obj	<i>Ridge objective function for model based optimization</i>
-----------	--

Description

This function generates an objective function for model-based optimization based on the cross-validated log-likelihood of a tramnet model with a ridge penalty only. It is not intended to be called by the user directly, instead it will be given as an argument to [mbo_tramnet](#).

Usage

```
ridge_obj(object, minlambda = 0, maxlambda = 16, folds, noisy = FALSE, fold)
```

Arguments

object	object of class tramnet
minlambda	minimum value for lambda (default: 0)
maxlambda	maximum value for lambda (default: 16)
folds	self specified folds for cross validation (mainly for reproducibility and comparability purposes)
noisy	indicates whether folds for k-fold cross-validation should be random for each iteration, leading to a noisy objective function (default: FALSE)
fold	fold for cross validation

Value

Single objective function for model based optimization of class [ObjectiveRFun](#).

simulate.tramnet	<i>simulate method for class "tramnet"</i>
------------------	--

Description

simulate method for class "tramnet"

Usage

```
## S3 method for class 'tramnet'  
simulate(object, nsim = 1, seed = NULL, newdata, bysim = TRUE, ...)
```

Arguments

object	object of class "tramnet"
nsim	number of simulation
seed	random number generator seed
newdata	new data to simulate from, defaults to the data used for fitting object
bysim	see simulate.ctm
...	Additional arguments to simulate.ctm

Value

returns a list of data.frames containing parametric bootstrap samples based on the data supplied in newdata

Author(s)

Lucas Kook

summary.tramnet	<i>summary method for class "tramnet"</i>
-----------------	---

Description

summary method for class "tramnet"

Usage

```
## S3 method for class 'tramnet'  
summary(object, ...)
```

Arguments

object	object of class "tramnet"
...	additional arguments

Value

Returns an object of class "summary.tramnet" containing information about the model, optimization status, sparsity and tuning parameters

Author(s)

Lucas Kook

tramnet

Regularised Transformation Models

Description

Partially penalized and constrained transformation models, including Cox models and continuous outcome logistic regression. The methodology is described in the tramnet vignette accompanying this package.

Usage

```
tramnet(model, ...)  
  
## Default S3 method:  
tramnet(model, x, lambda, alpha, constraints = NULL, ...)  
  
## S3 method for class 'formula'  
tramnet(  
  model,  
  data,  
  subset,  
  weights,  
  offset,  
  na.action = na.omit,  
  lambda,  
  alpha,  
  constraints = NULL,  
  ...  
)
```

Arguments

<code>model</code>	an object of class "tram" as returned by any of the modelling functions from package tram (default method) or a formula describing the model (formula method)
<code>...</code>	additional parameters to <code>psolve</code> (default method) or to <code>tram</code> (formula method), e.g., <code>distribution</code> , <code>transformation</code> , <code>order</code> or <code>log_first</code>
<code>x</code>	a numeric matrix, where each row corresponds to the same row in the data argument used to fit <code>model</code>
<code>lambda</code>	a positive penalty parameter for the whole penalty function
<code>alpha</code>	a mixing parameter (between zero and one) defining the fraction between absolute and quadratic penalty terms
<code>constraints</code>	an optional list containing a matrix of linear inequality constraints on the regression coefficients and a vector specifying the rhs of the inequality. In the formula method, the constraints refer to the coefficients on the original scale of the covariates
<code>data</code>	an optional data frame containing the variables in the model
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process
<code>weights</code>	an optional vector of case weights to be used in the fitting process
<code>offset</code>	an optional vector of offsets
<code>na.action</code>	a function which indicates what should happen when the data contain NAs

Details

The default method takes an unpenalized baseline model of class "tram", for instance `Coxph(y ~ 1, data = d)`, and a matrix of covariates `x`, which should be scaled by the user. The formula method fits the baseline model and sets up the design matrix in one go. It is the penalized analogue of `tram`: the model is fixed through the arguments `distribution` and `transformation` (and others) passed via `...` to `tram`. Convenience wrappers such as `Coxphnet` exist for the most common models.

In the formula method, the left-hand side of `model` (including strata, as in `y | s ~ x`) defines the baseline transformation, all terms on the right-hand side are penalized. The design matrix is obtained via `model.matrix` without intercept and its columns are scaled to unit variance. They are not centered, since the baseline transformation absorbs any shift in location. The scaling of the training data is stored and re-used whenever newdata is supplied, e.g., in `predict`, `logLik` or `cvl_tramnet`. Coefficients are reported on the original scale of the covariates, see `coef.tramnet`.

Value

An object of class "tramnet" with `coef`, `logLik`, `summary`, `simulate`, `residuals`, `predict`, `model.matrix` and `plot` methods

Author(s)

Lucas Kook, Balint Tamasi, Sandra Sigfried

Examples

```
if (require("penalized") & require("survival")) {  
  ## --- Comparison with penalized  
  data("nki70", package = "penalized")  
  nki70$resp <- with(nki70, Surv(time, event))  
  x <- scale(model.matrix(~ 0 + DIAPH3 + NUSAP1 + TSPYL5 + C20orf46,  
    data = nki70))  
  fit <- penalized(response = resp, penalized = x, lambda1 = 1, lambda2 = 0,  
    standardize = FALSE, data = nki70)  
  y <- Coxph(resp ~ 1, data = nki70, order = 10, log_first = TRUE)  
  fit2 <- tramnet(y, x, lambda = 1, alpha = 1) ## L1 only  
  coef(fit)  
  coef(fit2)  
  
  ## --- Formula interface, scaling is done internally  
  fit3 <- tramnet(resp ~ DIAPH3 + NUSAP1 + TSPYL5 + C20orf46, data = nki70,  
    lambda = 1, alpha = 1, distribution = "MinExtrVal",  
    transformation = "smooth", negative = FALSE, order = 10,  
    log_first = TRUE)  
  coef(fit3)  
  coef(fit3, scaled = TRUE)  
}
```

Index

- * **model**
 - Coxphnet, [4](#)
 - tramnet, [23](#)
- AcqFunction, [13](#)
- BoxCox, [4](#)
- BoxCoxnet (Coxphnet), [4](#)
- coef.tramnet, [2](#), [6](#), [24](#)
- coef.tramnet_Lm, [3](#)
- Colr, [4](#)
- Colrnet (Coxphnet), [4](#)
- Coxph, [4](#)
- Coxphnet, [4](#), [24](#)
- cvl_tramnet, [6](#), [7](#), [24](#)
- elnet_obj, [8](#)
- estfun.tramnet, [9](#)
- lasso_obj, [10](#)
- LearnerRegr, [13](#)
- Lehmann, [4](#)
- Lehmannnet (Coxphnet), [4](#)
- Lm, [4](#)
- Lmnet (Coxphnet), [4](#)
- logLik.tramnet, [11](#)
- mbo_recommended, [12](#)
- mbo_tramnet, [8](#), [10](#), [12](#), [12](#), [21](#)
- mlr_acqfunctions, [13](#)
- mlr_learners, [13](#)
- model.matrix, [6](#), [24](#)
- model.matrix.tramnet, [14](#)
- ObjectiveRFun, [9](#), [10](#), [21](#)
- OptimInstanceBatchSingleCrit, [13](#)
- plot, [15](#), [16](#)
- plot.tramnet, [15](#)
- plot_path, [16](#)
- predict.tramnet, [17](#)
- print.summary.tramnet, [17](#)
- print.tramnet, [18](#)
- prof_alpha, [18](#)
- prof_lambda, [19](#)
- psolve, [24](#)
- residuals.tramnet, [20](#)
- ridge_obj, [21](#)
- simulate.ctm, [22](#)
- simulate.tramnet, [22](#)
- summary, [18](#)
- summary.tramnet, [22](#)
- Survreg, [4](#)
- Survregnet (Coxphnet), [4](#)
- tram, [6](#), [24](#)
- tramnet, [4](#), [7](#), [23](#)