

Package: tramnet (via r-universe)

September 19, 2026

Title Penalized Transformation Models

Version 0.1-1

Date 2025-10-07

URL <http://ctm.R-forge.R-project.org>

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.), models for binary and ordered categorical variables, normal and transformed-normal (Box-Cox type) linear models, and continuous outcome logistic regression. Hyperparameter tuning is facilitated through model-based optimization functionalities from package 'mlr3MBO'. The methodology is described in Kook et al. (2021) <[doi:10.32614/RJ-2021-054](https://doi.org/10.32614/RJ-2021-054)>. 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. (2016) <[doi:10.48550/arXiv.1703.03373](https://doi.org/10.48550/arXiv.1703.03373)>, respectively.

Depends R (>= 4.0.0), tram (>= 1.0-0), CVXR (>= 1.0-0)

Imports mlt, basefun, sandwich, stats, paradox, mlr3mbo, bbotk

Suggests penalized, TH.data, survival, testthat (>= 3.0.0)

License GPL-2

RoxygenNote 7.3.1

Encoding UTF-8

Config/testthat/edition 3

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

Repository <https://r-forge.r-universe.dev>

Date/Publication 2026-05-22 09:46:49 UTC

RemoteUrl <https://github.com/r-forge/ctm>

RemoteRef HEAD

RemoteSha 6cd3bd0ce8570d012a6fdd3955b10c8140794b84

RemoteSubdir pkg/tramnet

Contents

cvl_tramnet	2
LmNET	3
logLik.tramnet	5
mbo_recommended	7
mbo_tramnet	8
plot.tramnet	9
plot_path	10
prof_alpha	11
prof_lambda	12
tramnet	13

Index	15
--------------	-----------

cvl_tramnet	<i>Cross-validating tramnet models</i>
-------------	--

Description

k-fold cross validation for "tramnet" objects over a grid of the tuning parameters based on out-of-sample log-likelihood.

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 hyper-parameters as an object of class "cvl_tramnet"

Examples

```

set.seed(241068)
if (require("survival") & require("TH.data")) {
  data("GBSG2", package = "TH.data")
  X <- 1 * matrix(GBSG2$horTh == "yes", ncol = 1)
  colnames(X) <- "horThyes"
  GBSG2$urv <- 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))
}

```

LmNET

*Regularized transformation model classes***Description**

Regularized transformation model classes

Usage

```

LmNET(
  formula,
  data,
  lambda = 0,
  alpha = 1,
  tram_args = NULL,
  constraints = NULL,
  ...
)

```

```

BoxCoxNET(
  formula,
  data,
  lambda = 0,
  alpha = 1,
  tram_args = NULL,
  constraints = NULL,
  ...
)

```

```

ColrNET(
  formula,
  data,

```

```
    lambda = 0,  
    alpha = 1,  
    tram_args = NULL,  
    constraints = NULL,  
    ...  
)
```

```
SurvregNET(  
  formula,  
  data,  
  lambda = 0,  
  alpha = 1,  
  tram_args = NULL,  
  constraints = NULL,  
  ...  
)
```

```
CoxphNET(  
  formula,  
  data,  
  lambda = 0,  
  alpha = 1,  
  tram_args = NULL,  
  constraints = NULL,  
  ...  
)
```

```
LehmannNET(  
  formula,  
  data,  
  lambda = 0,  
  alpha = 1,  
  tram_args = NULL,  
  constraints = NULL,  
  ...  
)
```

```
PolrNET(  
  formula,  
  data,  
  lambda = 0,  
  alpha = 1,  
  tram_args = NULL,  
  constraints = NULL,  
  ...  
)
```

Arguments

formula	Formula specifying the regression. See tram .
data	Object of class "data.frame" containing the variables referred to in the formula model.
lambda	A positive penalty parameter for the whole penalty function.
alpha	A mixing parameter (between zero and one) defining the fraction between lasso and ridge penalties, where $\alpha = 1$ corresponds to a pure lasso and $\alpha = 0$ to a pure ridge penalty.
tram_args	Additional arguments (besides model and data) passed to tram_fun.
constraints	An optional list containing a matrix of linear inequality constraints on the regression coefficients and a vector specifying the rhs of the inequality.
...	Additional arguments passed to solve .

Value

Object of class "tramnet".

logLik.tramnet	<i>S3 methods for class "tramnet"</i>
----------------	---------------------------------------

Description

S3 methods for class "tramnet"

Usage

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

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

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

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

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

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

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

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

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

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

Arguments

object	Object of class "tramnet".
parm	Parameters to evaluate the log likelihood at.
w	Optional vector of sample weights.
newdata	Data to evaluate the log likelihood at.
add_penalty	Whether or not to return the penalized log-likelihood (default add_penalty = FALSE).
...	Ignored.
with_baseline	If TRUE, also prints coefficients for the baseline transformation.
tol	Tolerance when an estimate should be considered 0 and not returned (default tol = 1e-6).
as.lm	See coef.mlt
nsim	Number of simulations, see simulate.mlt .
seed	Random seed, see simulate.mlt .
bysim	Return by simulation, see simulate.mlt .

x Object of class "tramnet".
 digits Number of digits to print.

Value

Returns (potentially weighted w) log-likelihood based on object evaluated at parameters parm and data newdata
 Numeric vector containing the model shift parameter estimates
 Numeric vector containing the linear model shift parameter estimates
 Vector of predictions based on object evaluated at each row of newdata
 Returns a list of data.frames containing parametric bootstrap samples of the response based on the data supplied in newdata
 Matrix of score contributions w.r.t. model parameters evaluated at parm
 Returns a numeric vector of residuals for each row in newdata
 Object of class "summary.tramnet".
 Object of class "summary.tramnet".
 Invisible x.

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

Description

Extracts the "optimal" tuning parameters from the output of mbo_tramnet and fits the corresponding tramnet model.

Usage

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

Arguments

mbo_obj Object returned by [mbo_tramnet](#).
 m0 Null model of class "tram", see [tramnet.tram](#).
 x Design matrix, see [tramnet.tram](#).
 ... Additional arguments to tramnet.

Value

Object of class "tramnet".

mbo_tramnet

Model based optimization for regularized transformation models

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_iter = 5,
  minlambda = 0,
  maxlambda = 16,
  minalpha = 0,
  maxalpha = 1,
  folds = NULL,
  noisy = FALSE,
  obj_type = c("lasso", "ridge", "elnet"),
  verbose = TRUE,
  ...
)
```

Arguments

object	Object of class "tramnet".
fold	Number of cross validation folds.
n_iter	Maximum number of iterations in model-based optimization routine.
minlambda	Minimum value for lambda (default minlambda = 0).
maxlambda	Maximum value for lambda (default maxlambda = 16).
minalpha	Minimum value for alpha (default minalpha = 0).
maxalpha	Maximum value for alpha (default maxalpha = 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 noisy = FALSE).
obj_type	Objective type, one of "lasso", "ridge" or "elnet".
verbose	Toggle for a verbose output (default verbose = TRUE)
...	Currently ignored.

Value

See [Optimizer](#)'s optimize function which returns a `data.table::data.table`.

plot.tramnet	<i>Plot "tramnet" objects</i>
--------------	-------------------------------

Description

Plot "tramnet" objects

Usage

```
## S3 method for class 'tramnet'
plot(
  x,
  newdata = NULL,
  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	See plot.ctm .
type	See plot.ctm .
q	See plot.ctm .
prob	See plot.ctm .
K	See plot.ctm .
col	See plot.ctm .
lty	See plot.ctm .
add	See plot.ctm .
...	Additional arguments passed to plot.ctm .

plot_path	<i>Plot regularization paths</i>
-----------	----------------------------------

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 plot_logLik = FALSE).
...	Additional arguments to plot

Value

None.

Examples

```
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)
}
```

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	Object of class "tramnet".
min_alpha	Minimal value of alpha (default min_alpha = 0).
max_alpha	Maximal value of alpha (default max_alpha = 1).
nprof	Number of profiling steps (default nprof = 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

Examples

```
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

model	Object of class "tramnet".
min_lambda	Minimal value of lambda (default min_lambda = 0).
max_lambda	Maximal value of lambda (default max_lambda = 15).
nprof	Number of profiling steps (default nprof = 5).
as.lm	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

Examples

```
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)
}
```

tramnet	<i>Regularized transformation models</i>
---------	--

Description

Regularized transformation models

Usage

```
tramnet(model, ...)

## S3 method for class 'formula'
tramnet(
  model,
  data,
  lambda,
  alpha,
  tram_fun,
  tram_args = NULL,
  constraints = NULL,
  groups = NULL,
  ...
)

## S3 method for class 'tram'
tramnet(model, x, lambda, alpha, constraints = NULL, groups = NULL, ...)
```

Arguments

model	Either a "formula" specifying the regression or an object of class "tram".
...	Additional arguments passed to solve .
data	Object of class "data.frame" containing the variables referred to in the formula model.
lambda	A positive penalty parameter for the whole penalty function.
alpha	A mixing parameter (between zero and one) defining the fraction between lasso and ridge penalties, where $\alpha = 1$ corresponds to a pure lasso and $\alpha = 0$ to a pure ridge penalty.
tram_fun	Character referring to an implementation in package 'tram'. See BoxCoxNET for the implemented models.
tram_args	Additional arguments (besides model and data) passed to tram_fun.
constraints	An optional list containing a matrix of linear inequality constraints on the regression coefficients and a vector specifying the rhs of the inequality.
groups	For group lasso penalties, groups can be supplied as a vector of consecutive integers of the same length as columns in x.
x	A numeric matrix, where each row corresponds to the same row in the data argument used to fit model.

Details

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.

Value

An object of class "tramnet" with coef, logLik, summary, simulate, residuals and plot methods

References

Lucas Kook and Torsten Hothorn, The R Journal (2021) 13:1, pages 581-594. doi:[10.32614/RJ-2021054](https://doi.org/10.32614/RJ-2021054)

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)  
}
```

Index

BoxCoxNET, [13](#)
BoxCoxNET (LmNET), [3](#)

coef.mlt, [6](#)
coef.tramnet(logLik.tramnet), [5](#)
coef.tramnet_Lm(logLik.tramnet), [5](#)
ColrNET (LmNET), [3](#)
CoxphNET (LmNET), [3](#)
cv1_tramnet, [2](#)

estfun.tramnet(logLik.tramnet), [5](#)

LehmannNET (LmNET), [3](#)
LmNET, [3](#)
logLik.tramnet, [5](#)

mbo_recommended, [7](#)
mbo_tramnet, [7](#), [8](#)

Optimizer, [9](#)

plot, [10](#)
plot.ctm, [9](#)
plot.tramnet, [9](#)
plot_path, [10](#)
PolrNET (LmNET), [3](#)
predict.tramnet(logLik.tramnet), [5](#)
print.summary.tramnet(logLik.tramnet),
 [5](#)
print.tramnet(logLik.tramnet), [5](#)
prof_alpha, [11](#)
prof_lambda, [12](#)

residuals.tramnet(logLik.tramnet), [5](#)

simulate.mlt, [6](#)
simulate.tramnet(logLik.tramnet), [5](#)
solve, [5](#), [13](#)
summary.tramnet(logLik.tramnet), [5](#)
SurvregNET (LmNET), [3](#)

tram, [5](#)

tramnet, [13](#)
tramnet.tram, [7](#)