Regularized Transformation Models: The tramnet Package

Lucas Kook[1] Torsten Hothorn[2]

[1]Copenhagen Business School [2]University of Zurich

tramnet version 0.1-0

Abstract

The tramnet package implements regularized linear transformation models by combining the flexible class of transformation models from tram with constrained convex optimization implemented in CVXR. Regularized transformation models unify many existing and novel regularized regression models under one theoretical and computational framework. Regularization strategies implemented for transformation models in tramnet include the LASSO, ridge regression and the elastic net and follow the parametrization in glmnet. Several functionalities for optimizing the hyperparameters, including model-based optimization based on the mlr3mbo package, are implemented. A multitude of S3 methods are deployed for visualization, handling and simulation purposes. This work aims at illustrating all facets of tramnet in realistic settings and comparing regularized transformation models with existing implementations of similar models.

This vignette is an updated version of Kook and Hothorn (2021), which describes an earlier version of tramnet. Since then, tramnet gained a formula interface (Section 1.5), which takes care of setting up and scaling the design matrix, and model-based optimization is now based on mlr3mbo.

Contents

1 Introduction **2**
- 1.1 Transformation models . . . 2
- 1.2 Regularization . . . 3
- 1.3 Constrained convex optimization . . . 3
- 1.4 Model-based optimization . . . 4
- 1.5 Basic usage . . . 5
- 1.6 Censoring and likelihood forms . . . 5

2 Prostate cancer data analysis **5**
- 2.1 Linear and Box-Cox type regression models . . . 6
- 2.2 Hyperparameter tuning . . . 8
  - 2.2.1 Model-based optimization . . . 10
  - 2.2.2 Regularization paths . . . 11
- 2.3 Additional constraints . . . 11

3 S3 Methods **12**

1 Introduction

A plethora of R packages exist to estimate generalized linear regression models via penalized maximum likelihood, such as penalized (Goeman, 2010) and glmnet (Friedman et al., 2010). Both packages come with an extension to fit a penalized form of the Cox proportional hazard model. The tramnet package aims at unifying the above mentioned and several novel models using the theoretical and computational framework of transformation models. Novel models in this class include Continuous Outcome Logistic Regression (COLR) as introduced by Lohse et al. (2017) and Box-Cox type regression models with a transformed conditionally normal response (Box and Cox, 1964; Hothorn et al., 2023).

The disciplined convex optimization package CVXR (Fu et al., 2020) is applied to solve the constrained convex optimization problems that arise when fitting regularized transformation models. Transformation models are introduced in Section 1.1, for a more theoretical treatise we refer to Hothorn et al. (2014, 2018); Hothorn (2020b). Convex optimization and domain specific languages are briefly discussed in Section 1.3, followed by a treatment of model-based optimization for hyperparameter tuning (1.4).

1.1 Transformation models

In stark contrast to penalized generalized linear models, regularized transformation models aim at estimating the response's whole conditional distribution instead of focusing on a single moment, *e.g.* the conditional mean. This conditional distribution function of a response Y is decomposed into an *a priori* chosen absolute continuous and log-concave error distribution F and a conditional transformation function $h(y|\boldsymbol{x}, \boldsymbol{s})$ that depends on the measured covariates $\boldsymbol{x}$ and stratum variables $\boldsymbol{s}$ and is monotone increasing in y. Although the model class is more flexible, packages tram and tramnet focus on stratified linear transformation models of the form

$$\mathbb{P}\left(Y \leq y | \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{S} = \boldsymbol{s}\right) = F\left(h(y|\boldsymbol{s}, \boldsymbol{x})\right) = F\left(h(y|\boldsymbol{s}) - \boldsymbol{x}^\top \boldsymbol{\beta}\right). \tag{1}$$

Here, the baseline transformation is allowed to vary with stratum variables $\boldsymbol{s}$, while covariate effects $\boldsymbol{\beta}$ are restricted to be shifts common to all baseline transformations $h(y|\boldsymbol{s})$.

In order for the model to represent a valid cumulative distribution function, $F\left(h(y|\boldsymbol{s}, \boldsymbol{x})\right)$ has to be monotone increasing in y and thus in h for all possible strata $\boldsymbol{s}$ and all possible configurations of the covariates $\boldsymbol{x}$. To ensure monotonicity, h is parametrized in terms of a basis expansion using Bernstein polynomials as implemented in the basefun package (Hothorn, 2020b). Hence, h is of the form

$$h(y) = \boldsymbol{a}_{\mathrm{Bs},p}(y)^\top \boldsymbol{\vartheta},$$

where $\boldsymbol{a}_{\mathrm{Bs},p}(y)$ denotes the vector of basis functions in y of order p and $\boldsymbol{\vartheta}$ are the coefficients for each basis function. Conveniently, $\boldsymbol{a}_{\mathrm{Bs},p}(y)^\top \boldsymbol{\vartheta}$ is monotone increasing in y as long as

$$\vartheta_i \leq \vartheta_{i+1} \quad \forall\, i = 0, \ldots, p-1 \tag{2}$$

holds. For the concrete parameterization of stratified linear transformation models the reader is referred to Hothorn et al. (2023).

Many contemporary models can be understood as linear transformation models, such as the normal linear regression model, logistic regression for binary, ordered and continuous responses, as well as exponential, Weibull and Rayleigh regression and the Cox model in survival analysis. Thus, by appropriately choosing and parametrizing F and h one can understand all those models in the same maximum likelihood-based framework. One can formulate the corresponding likelihood contributions not only for exact observations, but under any form of random censoring and truncation for continuous and discrete or ordered categorical responses.

Given a univariate response Y and a set of covariates $\boldsymbol{X}$ one can specify the following cumulative distribution function and density valid for any linear transformation model,

$$F_{Y|\boldsymbol{X}=\boldsymbol{x}}(y|\boldsymbol{s},\boldsymbol{x}) = F\left(h(y \mid \boldsymbol{s}) - \boldsymbol{x}^\top\boldsymbol{\beta}\right),$$
$$f_{Y|\boldsymbol{X}=\boldsymbol{x}}(y|\boldsymbol{s},\boldsymbol{x}) = F'\left(h(y \mid \boldsymbol{s}) - \boldsymbol{x}^\top\boldsymbol{\beta}\right) \cdot h'(y \mid \boldsymbol{s}).$$

From here, the log-likelihood contributions for exact, right, left, and interval censored responses can be derived as

$$\ell_i(\boldsymbol{\vartheta},\boldsymbol{\beta};y_i,\boldsymbol{s}_i,\boldsymbol{x}_i) = \begin{cases} \log\left(F'\left(h\left(y_i \mid \boldsymbol{s}_i\right) - \boldsymbol{x}_i^\top\boldsymbol{\beta}\right)\right) + \log\left(h'(y_i \mid \boldsymbol{s}_i)\right) & y_i & \text{exact} \\ \log\left(F\left(h(\bar{y} \mid \boldsymbol{s}_i) - \boldsymbol{x}_i^\top\boldsymbol{\beta}\right)\right) & y_i \in (-\infty,\bar{y}] & \text{left} \\ \log\left(1 - F\left(h(\underline{y} \mid \boldsymbol{s}_i) - \boldsymbol{x}_i^\top\boldsymbol{\beta}\right)\right) & y_i \in (\underline{y},\infty) & \text{right} \\ \log\left(F\left(h(\bar{y} \mid \boldsymbol{s}_i) - \boldsymbol{x}_i^\top\boldsymbol{\beta}\right) - F\left(h(\underline{y} \mid \boldsymbol{s}_i) - \boldsymbol{x}_i^\top\boldsymbol{\beta}\right)\right) & y_i \in (\underline{y},\bar{y}] & \text{interval.} \end{cases}$$

The joint log-likelihood of several observations $y_1,\ldots,y_n$ is obtained by summing over the individual log-likelihood contributions ℓ_i under the assumption that the individual samples are independent and identically distributed, the case exclusively dealt with by tramnet.

1.2 Regularization

The aim of tramnet is to enable the estimation of regularized stratified linear transformation models. This is achieved by optimizing a penalized form of the log-likelihood introduced in the last section. The penalized log-likelihood,

$$\tilde{\ell}(\boldsymbol{\vartheta},\boldsymbol{\beta},\lambda,\alpha;y,\boldsymbol{s},\boldsymbol{x}) = \ell(\boldsymbol{\vartheta},\boldsymbol{\beta};y,\boldsymbol{s},\boldsymbol{x}) - \lambda\left(\alpha\left\|\boldsymbol{\beta}\right\|_1 + \frac{1}{2}(1-\alpha)\left\|\boldsymbol{\beta}\right\|_2^2\right),$$

consists of the unpenalized log-likelihood and an additional penalty term. Note that only the shift parameters $\boldsymbol{\beta}$ are penalized, whereas the coefficients for the baseline transformation $\boldsymbol{\vartheta}$ remain unpenalized. The parameterization of the penalty is chosen to be the same as in glmnet, consisting of a global penalization parameter λ and a mixing parameter α controlling the amount of L_1 compared to L_2 penalization.

The two penalties and any combination thereof have unique properties and may be useful under different circumstances. A pure L_1 penalty was first introduced by Tibshirani (1996) in an OLS framework and was dubbed the LASSO (Least Absolute Shrinkage and Selection Operator) due to its property of shrinking regression coefficients exactly to 0 for large enough λ. A pure LASSO penalty can be obtained in a regularized transformation model by specifying $\alpha = 1$. Applying an L_2 penalty in an OLS problem was introduced more than five decades earlier by Tikhonov (1943) and later termed ridge regression (Hoerl and Kennard, 1970). In contrast to LASSO, ridge regression leads to shrunken regression coefficients, but does not perform automatic variable selection. Zou and Hastie (2005) picked up on both approaches, discussed their advantages, disadvantages and overall characteristics and combined them into the elastic net penalty, a convex combination of an L_1 and L_2 penatly controlled by the mixing parameter α. Some of these properties will be illustrated for different models and a real world data set in sections 1.6 and 2.2.

1.3 Constrained convex optimization

Special algorithms were developed to optimize regularized objective functions, most prominently the LARS and LARS-EN algorithm (Efron et al., 2004) and variants thereof for the penalized Cox model (Goeman, 2010). However, the aim of tramnet is to solve the objective functions arising in regularized transformation models in a single computational framework. Due to the

log-concavity of all choices for F in this package and $h(y)$ being monotone increasing in y, the resulting log-likelihood contributions for any form of censoring and truncation are concave and can thus be solved by constrained convex optimization.

The fairly recent development of CVXR allows the specification of constrained convex optimization problems in terms of domain-specific language, yielding an intuitive and highly flexible framework for constrained optimization. Because checking convexity of an aribitrarily complex expression is extremely hard, CVXR makes use of a library of smaller expressions, called atoms, with known monotonicity and curvature and tries to decompose the objective at hand using a set of rules from disciplined convex optimization (DCP) (Grant et al., 2006). Thus a complex expression's curvature can be more easily determined.

More formally, convex optimization aims at solving a problem of the form

$$\begin{aligned} \underset{\boldsymbol{\vartheta}}{\text{minimize}} \quad & g(\boldsymbol{\vartheta}) \\ \text{subject to} \quad & g_i(\boldsymbol{\vartheta}) \leq 0, \; i = 1, \dots, K \\ & \boldsymbol{A}\boldsymbol{\vartheta} = \boldsymbol{b}, \end{aligned}$$

where $\boldsymbol{\vartheta} \in \mathbb{R}^p$ is the parameter vector, $g(\boldsymbol{\vartheta})$ is the objective function to be optimized, $g_i(\boldsymbol{\vartheta})$ specify the inequality constraints and $\boldsymbol{A} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{b} \in \mathbb{R}^p$ parametrize any equality constraints on $\boldsymbol{\vartheta}$. Importantly, the objective function and all inequality constraint functions are convex (Boyd and Vandenberghe, 2004).

The likelihood $\sum_i \ell_i(\boldsymbol{\vartheta}, \boldsymbol{\beta}; y_i, \boldsymbol{s}_i, \boldsymbol{x}_i)$ for transformation models of the form (1) are convex for error distributions with log-concave density, because log-convexity of $-F'$ ensures the existence and uniqueness of the most likely transformation $\hat{h}(y)$ and the convexity of $-\ell(h; y, \boldsymbol{x})$. Because the penalty term

$$\lambda \left(\alpha \left\| \boldsymbol{\beta} \right\|_1 + \frac{1}{2}(1 - \alpha) \left\| \boldsymbol{\beta} \right\|_2^2 \right)$$

is convex in $\boldsymbol{\beta}$, it can be added to the negative log-likelihood while conserving convexity. However, monotonicity of h imposes inequality constraints on the parameters of the baseline transformation as illustrated in equation (2). The elegance of domain-specific language based optimizers comes to play when adding these and potential other inequality or equality constraints to the objective function, which will be showcased in Section 2.3. Thus, the optimisation routines implemented in package CVXR can be applied for computing maximum likelihood estimates of the parameters of model (1).

1.4 Model-based optimization

The predictive capabilities of regularized regression models heavily depend on the hyperparameters α and λ. Hyperparameter tuning can be addressed by a multitude of methods with varying computational complexity, advantages and disadvantages. Naive or random grid search for more than one tuning parameter are computationally demanding, especially if the objective function is expensive to evaluate. Model-based optimization circumvents this issue by fitting a surrogate model, usually a Gaussian process, to the objective function. The objective function is evaluated at an initial, *e.g.* a random latin hypercube, design, to which the Gaussian process is subsequently fit. The surrogate model then proposes the next set of hyperparameters to evaluate the objective function at by some infill criterion (Horn and Bischl, 2016). Bischl et al. (2017) implement model-based optimization for multi-objective blackbox functions in the mlrMBO package, which has been superseded by mlr3mbo, on which tramnet builds. The objective function can in theory be vector-valued and the tuning parameter spaces may be categorical. In tramnet the objective function is the cross-validated log-likelihood optimized using a Kriging surrogate model with expected improvement as the infill criterion. Model-based optimization for hyperparameter tuning is illustrated in section 2.

1.5 Basic usage

The initial step is fitting a potentially stratified, transformation model of the form

```
R> m1 <- tram(y | s ~ 1, ...)
```

omitting all explanatory variables. This sets up the basis expansion for the transformation function, whose regression coefficients will not be penalized, as mentioned in section 1.2. Additionally, `tramnet()` needs a model matrix including the predictors, whose regression coefficients ought to be penalized. For numerical reasons it is useful to provide a scaled model matrix, instead of the original data, such that every parameter is equally affected by the regularization. Lastly, `tramnet()` will need the tuning parameters $\alpha \in [0, 1]$ and $\lambda \in \mathbb{R}^+$, with α representing a mixing parameter and λ controlling the extent of regularization. Setting $\lambda = 0$ will result in an unpenalized model, regardless of the value of α.

```
R> x <- model.matrix(~ 0 + x, ...)
R> x_scaled <- scale(x)
R> mt <- tramnet(model = m1, x = x_scaled, lambda, alpha, ...)
```

Alternatively, the model can be specified via a formula. The formula method of `tramnet()` is the regularized analogue of `tram()`: the model class is fixed through the arguments `distribution` and `transformation` (and others) passed on to `tram()`. The left-hand side of the formula, including strata, defines the unpenalized baseline transformation and all terms on the right-hand side are penalized.

```
R> mt <- tramnet(y | s ~ x, data = d, lambda, alpha,
+                distribution = "Logistic", transformation = "smooth")
```

For the most common models, `Coxphnet()`, `Colrnet()`, `Lmnet()`, `BoxCoxnet()`, `Lehmannnet()` and `Survregnet()` are available as regularized counterparts of the corresponding functions in tram, *e.g.*

```
R> mt <- Colrnet(y | s ~ x, data = d, lambda, alpha)
```

Here, the design matrix is set up internally via `model.matrix()` and its columns are scaled to unit variance. Centering is not necessary, because the baseline transformation absorbs any shift in location. The scaling of the training data is stored and re-used whenever new data are supplied, *e.g.* in `predict()`, `logLik()` or in cross-validation, where the scaling is computed in each training fold anew. Regression coefficients are reported on the original scale of the covariates by default. S3 methods accompanying the `"tramnet"` class will be discussed in section 3.

1.6 Censoring and likelihood forms

Specific combinations of F and the form of censoring yield log-log-concave log-likelihoods. Under these circumstances tramnet is not yet able to solve the resulting optimization problem. Table 1 indicates which model class can be fitted under what type of censoring in the current version of tramnet.

2 Prostate cancer data analysis

The regularized normal linear and extensions to transformed normal regression models will be illustrated using the `Prostate` data set (Stamey et al., 1989), which was used by Zou and Hastie (2005) to highlight properties of the elastic net.

Table 1: Combinations of model classes and censoring types that are possible in the tramnet package. Due to missing disciplined geometric optimization rules in CVXR not every combination of error distribution and censoring type yield solvable objective functions. This will change with coming updates in the CVXR package.

Model Class	**Censoring Type**			
	Exact	**Left**	**Right**	**Interval**
BoxCox	✓	✗	✗	✗
Colr	✓	✓	✓	✗
Coxph	✓	✗	✓	✗
Lehmann	✓	✓	✗	✗

```
R> load("Prostate.rda")
R> Prostate$psa <- exp(Prostate$lpsa)
R> Prostate[, 1:8] <- scale(Prostate[, 1:8])
```

The data set contains 97 observations and 9 covariates. In the original paper the authors chose the log-transformed prostate specific antigen concentration (`lpsa`) as the response and used the eight remaining predictors log cancer volume (`lcavol`), log prostate weight (`lweight`), age of the patient (`age`), log benign prostatic hyperplasia amount (`lbph`), seminal vesicle invasion (`svi` coded as 1 for yes, 0 for no), log capsular penetration (`lcp`), Gleason score (`gleason`) and percentage Gleason score 4 or 5 (`pgg45`) as covariates.

2.1 Linear and Box-Cox type regression models

Zou and Hastie (2005) imposed an assumption on the conditional distribution of the response by log-transforming and fitting a linear model. In the following it is shown that the impact of this assumption may be assessed by estimating the baseline transformation from the data, followed by a comparison with the log-transformation applied by Zou and Hastie (2005). The linear models in `lpsa` and log(`psa`) are compared to transformation models with basis expansions in both log(`psa`) and `psa`, while specifying conditional normality of the transformed response. Additionally, the models are compared to an alternative implementation of regularized normal linear regression in penalized. Five different models will be used to illustrate important facettes of transformation models, including parametrization and interpretation. The models are summarized in Table 2 and will be elaborated on throughout this section. The comparison is based on unpenalized models first. Later, the section highlights the penalized models together with hyperparameter tuning.

```
R> fm_Pr <- psa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45
R> fm_Pr1 <- update(fm_Pr, ~ 0 + .)
R> x <- model.matrix(fm_Pr1, data = Prostate)
```

The normal linear regression model is implemented in tram's `Lm()` function. `Lm()`'s parametrization differs from the usual linear model, hence caution has to be taken when interpreting the resulting regression coefficients $\boldsymbol{\beta}$. In order to compare the results to an equivalent, already existing implementation, the same model is fitted using penalized.

```
R> m0 <- Lm(lpsa ~ 1, data = Prostate)
R> mt <- tramnet(m0, x = x, alpha = 0, lambda = 0)
R> mp <- penalized(response = Prostate$lpsa, penalized = x,
+                  lambda1 = 0, lambda2 = 0)
```

Table 2: Summary of the five models illustrated in section 2, including their name throughout the manuscript, the R code to fit them and the mathematical formulation of their conditional cumulative distribution function. For comparison `mp` is included as an ordinary linear model, which is equivalent to model `mt` in terms of log-likelihood, but differs in the parametrization of the transformation function h and thus yields scaled coefficient estimates (*cf.* Table 3). Model `mtp` is a linear model parametrized in terms of a Bernstein basis of maximum order 1. This will yield the same coefficient estimates as `mt` but a log-likelihood that is comparable to models `mt1` and `mt2`, whose transformation functions are parametrized in terms of higher order Bernstein bases. The `log_first` argument specifies whether the basis expansion is calculated on the log-transformed or untransformed response.

Name	**Code**	**Model for** $F_{Y\|\boldsymbol{X}=\boldsymbol{x}}(y\|\boldsymbol{x})$
`mp`	`penalized(response = lpsa, penalized = x)`	$\Phi\left(\vartheta_1 + \vartheta_2 \log(y) - \boldsymbol{x}^\top\boldsymbol{\beta}\right)$
`mt`	`Lm(lpsa ~ .)`	$\Phi\left(\vartheta_1 + \vartheta_2 \log(y) - \boldsymbol{x}^\top\boldsymbol{\beta}\right)$
`mtp`	`BoxCox(psa ~ ., log_first = TRUE, order = 1)`	$\Phi\left(\boldsymbol{a}_{\text{Bs},1}(\log(y))^\top\boldsymbol{\vartheta} - \boldsymbol{x}^\top\boldsymbol{\beta}\right)$
`mt1`	`BoxCox(psa ~ ., log_first = TRUE, order = 7)`	$\Phi\left(\boldsymbol{a}_{\text{Bs},7}(\log(y))^\top\boldsymbol{\vartheta} - \boldsymbol{x}^\top\boldsymbol{\beta}\right)$
`mt2`	`BoxCox(psa ~ ., log_first = FALSE, order = 11)`	$\Phi\left(\boldsymbol{a}_{\text{Bs},11}(y)^\top\boldsymbol{\vartheta} - \boldsymbol{x}^\top\boldsymbol{\beta}\right)$

A linear model of the form

$$Y = \tilde{\alpha} + \boldsymbol{x}^\top\tilde{\boldsymbol{\beta}} + \varepsilon, \quad \varepsilon \sim \mathrm{N}(0, \sigma^2)$$

can be understood as a transformation model through reparametrization as

$$\mathbb{P}(Y \leq y|\boldsymbol{X} = \boldsymbol{x}) = \Phi\left(\vartheta_1 + \vartheta_2 y - \boldsymbol{x}^\top\boldsymbol{\beta}\right).$$

Here $\vartheta_1 = -\tilde{\alpha}/\sigma$ is a reparametrized intercept term, $\vartheta_2 = 1/\sigma$ is the slope of the baseline transformation and the regression coefficients $\boldsymbol{\beta} = \tilde{\boldsymbol{\beta}}/\sigma$ represent scaled shift terms, influencing only the intercept. To recover the usual parametrization `tramnet::coef.Lm()` offers the `as.lm = TRUE` argument.

```
R> cfx_tramnet <- coef(mt, as.lm = TRUE)
```

The same model can be fitted using the formula interface. Because the covariates in `Prostate` have been standardized already, both interfaces lead to the same estimates.

```
R> fm_lpsa <- lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45
R> mf <- Lmnet(fm_lpsa, data = Prostate, alpha = 0, lambda = 0)
R> max(abs(coef(mf, as.lm = TRUE) - cfx_tramnet))
[1] 5.57e-12
```

With the formula interface, out-of-sample evaluation does not require any manual pre-processing of the test data, because the scaling of the training data is applied automatically.

```
R> idx <- seq_len(70)
R> mf_trn <- Lmnet(fm_lpsa, data = Prostate[idx, ], alpha = 0.5, lambda = 5)
R> logLik(mf_trn, newdata = Prostate[-idx, ])
'log Lik.' -89.2 (df=NA)
```

The transformation function for the linear model is depicted in Figure 1 (pink line). Because a linear baseline transformation imposes restrictive assumptions on the response's conditional distribution, it is advantageous to replace the linear baseline transformation by a more flexible

one. In the case of the Box-Cox type regression model the linear baseline transformation $h(y) = \vartheta_1 + \vartheta_2 \log y$ is replaced by the basis expansion $h(y) = \boldsymbol{a}_{\mathrm{Bs},7}(\log y)^\top \boldsymbol{\vartheta}$.

```
R> ord <- 7 # flexible baseline transformation
R> m01 <- BoxCox(psa ~ 1, data = Prostate, order = ord,
+                extrapolate = TRUE, log_first = TRUE)
R> mt1 <- tramnet(m01, x = x, alpha = 0, lambda = 0)
```

The Box-Cox type regression model is then estimated with the `BoxCox()` function, while specifying the appropriate maximum order of the Bernstein polynomial. Because, the more flexible transformation slightly deviates from being linear, the normal linear model yields a smaller log-likelihood (*cf.* Table 3). To make sure that this improvement is not due to the increased number of parameters and hence overfitting, the models predictive capacities could be compared via cross-validation.

These results hold for the pre-specified log transformation of the response and a basis expansion thereof. Instead of prespecifying the log-transformation, its 'logarithmic nature' can be estimated from the data. Afterwards one can compare the deviation from a log-linear baseline transformation graphically and by inspecting the predictive performance of the model in terms of the out-of-sample log-likelihood.

```
R> m02 <- BoxCox(psa ~ 1, order = 11, data = Prostate, extrapolate = TRUE)
R> mt2 <- tramnet(m02, x = x, lambda = 0, alpha = 0)
```

Indeed the baseline transformation in Figure 1 is similar to the basis expansion in the log-transformed response upon visual inspection. Because `mt` is estimated using the log-transformed response and `mt1` and `mt2` are based on the original scale of the response, the resulting model log-likelihoods are not comparable. To overcome this issue, one can fit a Box-Cox type model with maximum order 1, as this results in a linear, but alternatively parametrized baseline transformation.

```
R> m0p <- BoxCox(psa ~ 1, order = 1, data = Prostate, log_first = TRUE)
R> mtp <- tramnet(m0p, x = x, lambda = 0, alpha = 0)
```

Figure 1 plots the three distinct baseline transformations resulting from models `mt`, `mt1` and `mt2`. The initial assumption to model the prostate specific antigen concentration linearily on the log-scale seems to be valid when comparing the three transformation functions. The linear transformation in `lpsa` used in `mt` and the basis expansion in log(`psa`) (`mt1`) are almost indistinguishable and yield very similar coefficient estimates, as well as log-likelihoods (*cf.* Table 3, `mtp` *vs.* `mt1`). The basis expansion in `psa` (`mt2`) is expected to be less stable due to the highly skewed untransformed response. This is reflected in Figure 1, where the baseline transformation deviates from being linear towards the bounds of the response's support. However, the log-linear behaviour of h was clearly captured by this model and further supports the initial assumption of conditional log-normality of the response. For the same reasons, the resulting log-likelihood of `mt2` is smaller than for `mt1` (Table 3). Taken together, this exemplary analysis highlights the flexibility and usefulness of transformation models to judge crucial modelling assumptions.

2.2 Hyperparameter tuning

This section features cross-validation, model-based optimization and profiling functions for hyperparameter tuning, whose appropriate values are highly problem-dependent and hard to know in advance. tramnet implements naive grid search and model-based optimization in the functions `cvl_tramnet()` and `mbo_tramnet()`, respectively. In the framework of regularized transformation models it is very natural to choose the out-of-sample log-likelihood as the objective function, because the notion of a mean square loss does not make sense for survival, let alone censored

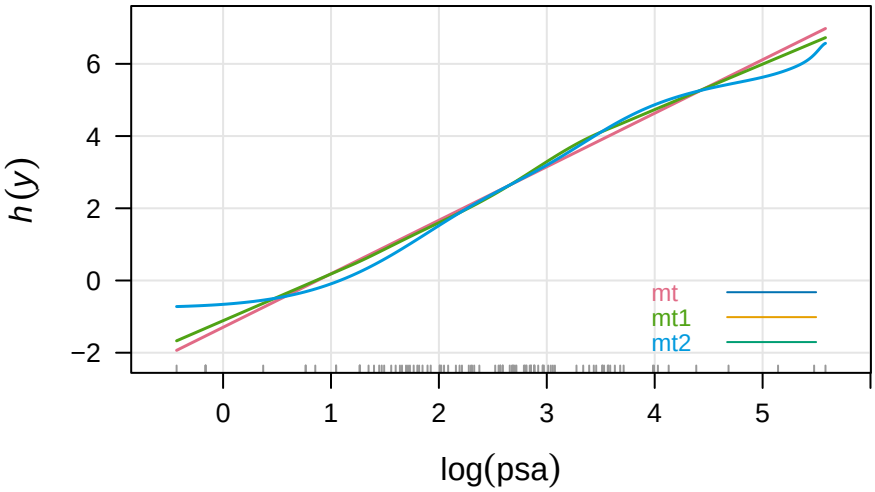


Figure 1: Comparison of different choices for the baseline transformation of the response (prostate specific antigen concentration) in the `Prostate` data. The original analysis prespecified a log-transformation of the response and then assumed conditional normality on this scale. Hence the baseline transformation of `mt` is of the form: $h(\text{lpsa}) = \vartheta_1 + \vartheta_2 \cdot \text{lpsa}$. Now one can allow a more flexible transformation function in log(`psa`) to judge any deviations of $h(\log(\text{psa}))$ from linearity, leading to a baseline transformation in terms of basis functions: $\boldsymbol{a}_{\text{Bs},7}(\log(\text{psa}))^\top \boldsymbol{\vartheta}$ in `mt1`. Lastly, instead of presuming a log-transformation one could estimate the baseline transformation from the raw response (`psa`), i.e. $h(\text{psa}) = \boldsymbol{a}_{\text{Bs},11}(\text{psa})^\top \boldsymbol{\vartheta}$ in `mt2`. In this case, a higher order basis expansion was chosen to account for the skewness of the raw response. Notably, all three baseline transformations are fairly comparable. The basis expansion in `psa` deviates from being log-linear towards the boundaries of the response's support, as there are only few observations.

outcomes. The out-of-sample log-likelihood is in fact the log score, which is a proper scoring rule (Gneiting and Raftery, 2007).

```
R> m0 <- BoxCox(lpsa ~ 1, data = Prostate, order = 7, extrapolate = TRUE)
R> mt <- tramnet(m01, x = x, alpha = 1, lambda = 0)
```

tramnet offers cross-validation in `cvl_tramnet()`, comparable to the `optL1()` and `optL2()` functions in penalized, which takes a sequence of values for λ and α and performs a simple – and arguably slow – grid search. Per default it computes 2-fold cross-validation, the user is encouraged, however, to judge the resulting bias-variance trade-off accordingly.

```
R> lambdas <- c(0, 10^seq(-4, log10(15), length.out = 4))
R> cvlt <- cvl_tramnet(object = mt, fold = 2, lambda = lambdas, alpha = 1)
```

In order to compare cross-validation across multiple packages and functions it is also possible to supply the folds for each row in the design matrix as a numeric vector, as for example returned by `penalized::optL1()`.

```
R> pen_cvl <- optL1(response = lpsa, penalized = x, lambda2 = 0, data = Prostate,
+                   fold = 2)
R> cvlt <- cvl_tramnet(object = mt, lambda = lambdas, alpha = 1,
+                      folds = pen_cvl$fold)
```

Table 3: Comparison of the three transformation models on the `Prostate` data. Coefficient estimates are shown for each model, together with the in-sample log-likelihood in the last column. The first three models, `mp`, `mt` and `mtp` use a linear baseline transformation in `lpsa` and log(`psa`), respectively. The `mp` model was fit using `penalized` and gives the scaled version of the regression coefficients in `mt`, but the same log-likelihood. At the same time, `mt` and `mtp` differ only in their response variable and its subsequent log-transformation in `mtp`, yielding the same coefficient estimates but a different log-likelihood. Models `mt1` and `mt2` allow a flexible basis expansion in log(`psa`) and `psa`, respectively. Model `mt1`, allowing for a flexible basis expansion in `lpsa`, fits the data the best, however the resulting coefficient estimates are similar for all models.

Model	**Coefficient estimates**								**logLik**
	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	
`mp`	0.69	0.23	-0.15	0.16	0.32	-0.15	0.03	0.13	-99.5
`mt`	1.03	0.33	-0.22	0.23	0.47	-0.22	0.05	0.19	-99.5
`mtp`	1.03	0.33	-0.22	0.23	0.47	-0.22	0.05	0.19	-339.9
`mt1`	1.03	0.34	-0.21	0.22	0.48	-0.23	0.04	0.21	-337.9
`mt2`	0.99	0.33	-0.20	0.20	0.50	-0.25	0.06	0.22	-345.1

The resulting object is of class `"cvl_tramnet"` and contains a table for the cross-validated log-likelihoods for each fold and the sum thereof, the 'optimal' tuning parameter constellation which resulted in the largest cross-validated log-likelihood, tables for the cross-validated regularization paths, the folds and lastly the full fit based on the 'optimal' tuning parameters (if `fit_opt = TRUE`). The cross-validated regularization paths highly depend on the chosen folds and the user is referred to the full profiling functions discussed in section 2.2.2.

2.2.1 Model-based optimization

In contrast to naive grid search, model-based optimization comprises more elegant methods for hyperparameter tuning. `tramnet` offers the `mbo_tramnet()` and `mbo_recommended()` functions. The former implements Kriging-based hyperparameter tuning for the elastic net, the LASSO and ridge regression. `mbo_tramnet()` takes a `"tramnet"` object as input and computes the cross-validated log-likelihood based on the provided `fold` or `folds` arugment. The initial design is a random latin hypercube design with `n_design` rows. The number of sequential fits of the surrogate models is specified through `n_iter` and the range of the tuning parameters can be specified by `max/min` arguments. The default infill criterion is expected improvement. However, Bischl et al. (2017) encourage the use of the lower confidence bound over expected improvement, which can be achieved in `mbo_tramnet()` by specifying `opt_crit = "cb"`. 10-fold cross-validation is used to compute the objective function for the initial design and each iteration. The recommended model is then extracted using `mbo_recommended()`.

```
R> tmbo <- mbo_tramnet(mt, obj_type = "elnet", fold = 10)
R> mtmbo <- mbo_recommended(tmbo, m0, x)
```

Unlike in the previous section, one can directly optimize the tuning parameters in an elastic net problem instead of optimizing over one hyperparameter at a time or having to specify LASSO or ridge regression *a priori*. The output of `mbo_tramnet()` is quite verbose and can be shortened by using the helper function `print_mbo()`.

```
R> print_mbo(tmbo)
Recommended parameters:
lmb = 7.565; alp = 0.433
Objective: y = 221
```

Interpreting the output, model-based optimization suggests a model with $\alpha = 0.43$ and $\lambda = 7.56$. This result stresses the advantages of model-based optimization over naive or random grid search in terms of complexity and computational efficiency.

```
R> coef(mtmbo)
 lcavol lweight     age    lbph     svi gleason   pgg45
 0.8192  0.2842 -0.0823  0.1410  0.3584  0.0239  0.1171
R> summary(mtmbo)$sparsity
[1] "8 regression coefficients, 7 of which are non-zero"
```

2.2.2 Regularization paths

As discussed before, it may be useful to inspect the full regularization paths over the tuning parameters λ and α. Akin to the functions `profL1()` and `profL2()` in package penalized, tramnet offers `prof_lambda()` and `prof_alpha()`. Since these functions take a fitted model of class `"tramnet"` as input, which is internally updated it is crucial to correctly specify the other tuning parameter in the model fitting step. In the example to come, `mt` was fit using $\alpha = 1$ and $\lambda = 0$ resulting in a LASSO penalty only when profiling over λ. The resulting profile is depicted in Figure 2.

```
R> pfl <- prof_lambda(mt)
```

`prof_lambda()` takes `min_lambda`, `max_lambda` and `nprof` as arguments and internally generates an equi-spaced sequence from `min_lambda` to `max_lambda` on the log scale of length `nprof`. By default this sequence ranges from 0 to 15 and is of length 5.

```
R> plot_path(pfl, plot_logLik = FALSE, las = 1, col = coll)
```

2.3 Additional constraints

In some applications, the specification of additional constraints on the shift parameters $\boldsymbol{\beta}$ are of interest. Most commonly, either positivity or negativity for some or all regression coefficients is aimed at. In tramnet additional inequality constraints can be specified via the `constraints` argument, which are internally handled as `constraints[[1]] %*% beta > constraints[[2]]`. Hence, to specify the constraint of strictly positive regression coefficients, one would supply an identity matrix of dimension p for the left hand side and the zero p-vector for the left hand side, as done in the following example.

```
R> m0 <- BoxCox(lpsa ~ 1, data = Prostate, extrapolate = TRUE)
R> mt <- tramnet(m0, x, alpha = 0, lambda = 0, constraints = list(diag(8),
+                                                                 rep(0, 8)))
R> coef(mt)
 lcavol lweight    lbph     svi gleason   pgg45
0.91263 0.30415 0.15590 0.40005 0.00909 0.11629
```

The coefficients with a negative sign in the model without additional positivity constraints now shrink to zero and the other coefficient estimates change as well.

```
R> summary(mt)$sparsity
[1] "8 regression coefficients, 6 of which are non-zero"
```

One can compare this model to the implementation in tram, where it is also possible to specify linear inequality constraints on the regression coefficients $\boldsymbol{\beta}$. Here, it is sufficient to specify

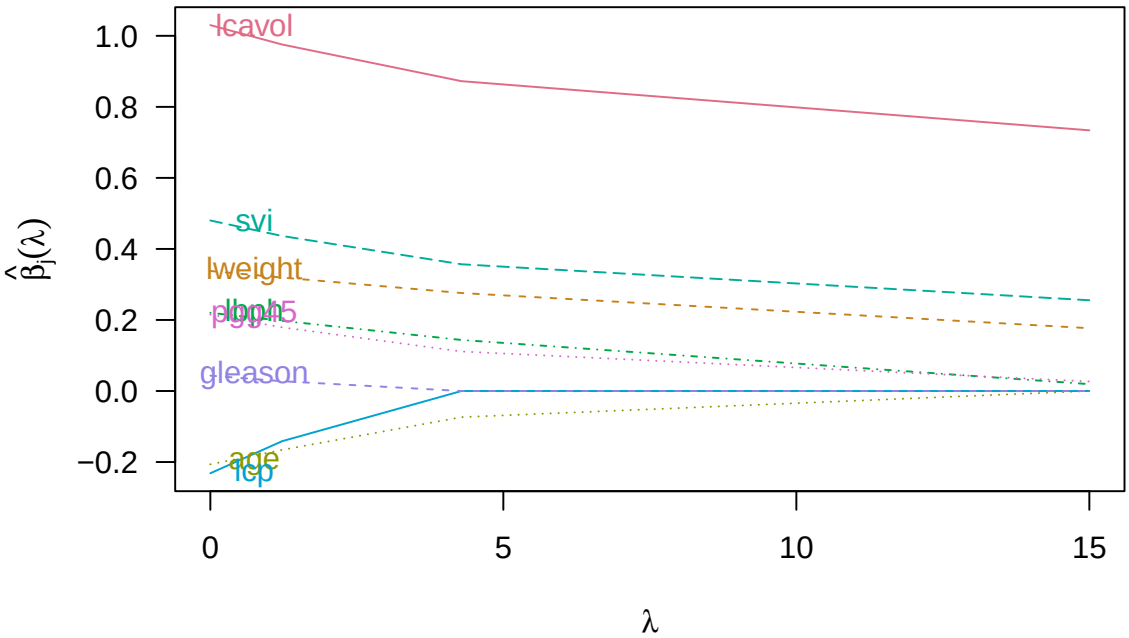


Figure 2: Full regularization paths for the tuning parameter λ using the default values of `plot_path()`.

`constraints = c("age >= 0", "lcp >= 0")` for the two non-positive coefficient estimates, due to the convexity of the underlying problem.

```
R> m <- BoxCox(lpsa ~ . - psa, data = Prostate, extrapolate = TRUE,
+               constraints = c("age >= 0", "lcp >= 0"))
R> max(abs(coef(m) - coef(mt, tol = 0)))
[1] 1.94e-06
```

Indeed, both optimizers arrive at virtually the same coefficient estimates.

3 S3 Methods

Building on the S3 infrastructure of the packages mlt and tram, this package provides corresponding methods for the following generics: `coef()`, `logLik()`, `plot()`, `predict()`, `simulate()`, `residuals()`, `estfun()` and `model.matrix()`. The methods' additional `"tramnet"`-specific arguments will be briefly discussed in this section.

`coef.tramnet()` suppresses the baseline transformation's coefficient estimates $\hat{\boldsymbol{\vartheta}}$ by default and considers shift parameter estimates $\hat{\boldsymbol{\beta}}$ below 10^{-6} as 0 to stress the selected variables only. This threshold can be controlled by the `tol` argument. Hence, `coef(mt, with_baseline = TRUE, tol = 0)` returns all coefficients. For models fitted via the formula interface, `coef(mt, scaled = TRUE)` returns the shift parameters on the scale of the internally scaled covariates, on which the penalty acts, and `model.matrix(mt, newdata)` returns the correspondingly scaled design matrix.

```
R> coef(mtmbo, with_baseline = TRUE, tol = 0)
Bs1(lpsa) Bs2(lpsa) Bs3(lpsa) Bs4(lpsa) Bs5(lpsa) Bs6(lpsa) Bs7(lpsa)
```

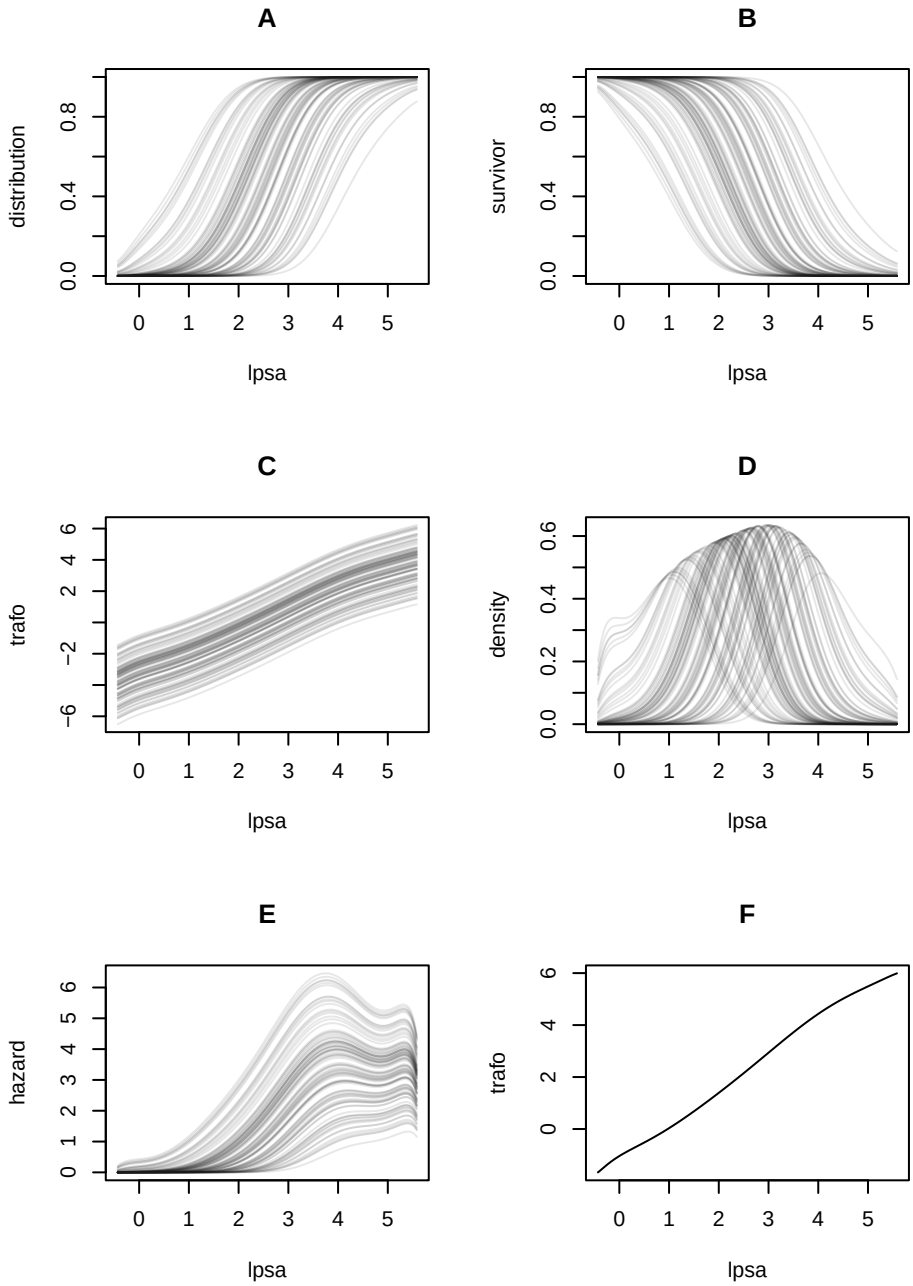
```
-3.82e+00 -3.32e+00 -1.58e+00 -1.58e+00  1.33e+00  1.46e+00  3.35e+00
Bs8(lpsa)    lcavol   lweight       age      lbph       svi       lcp
 3.84e+00  8.19e-01  2.84e-01 -8.23e-02  1.41e-01  3.58e-01 -2.72e-10
  gleason     pgg45
 2.39e-02  1.17e-01
```

The `logLik.tramnet()` method allows the log-likelihoods re-computation under new data (i.e. out-of-sample) and different coefficients (`parm`) and weights (`w`), as illustrated below. For new data, the out-of-sample log-likelihood is returned without subtracting the penalty.

```
R> logLik(mtmbo)
'log Lik.' -106 (df=NA)
R> cfx <- coef(mtmbo, with_baseline = TRUE, tol = 0)
R> cfx[5:8] <- 0.5
R> logLik(mtmbo, parm = cfx)
'log Lik.' -Inf (df=NA)
R> logLik(mtmbo, newdata = Prostate[1:10,])
'log Lik.' -14.8 (df=NA)
R> logLik(mtmbo, w = runif(n = nrow(mtmbo$x)))
'log Lik.' -63.9 (df=NA)
```

In the spirit of mlt's plotting methods for classes `"mlt"` and `"ctm"`, `plot.tramnet()` offers diverse plotting options for objects of class `"tramnet"`. The specification of new data and the type of plot is illustrated in the follwing code chunk and Figure 3.

```
R> par(mfrow = c(3, 2)); K <- 3e2
R> plot(mtmbo, type = "distribution", K = K, main = "A") # A, default
R> plot(mtmbo, type = "survivor", K = K, main = "B") # B
R> plot(mtmbo, type = "trafo", K = K, main = "C") # C
R> plot(mtmbo, type = "density", K = K, main = "D") # D
R> plot(mtmbo, type = "hazard", K = K, main = "E") # E
R> plot(mtmbo, type = "trafo", newdata = Prostate[1, ], col = 1, K = K, main = "F") # F
```



The `predict.tramnet()` method works in the same way as `predict.mlt()` and as such supports the types `trafo, distribution, survivor, density, logdensity, hazard, loghazard, cumhazard` and `quantile`. For `type = "quantile"` the corresponding probabilities (`prob`) have to be supplied as an argument to evaluate the quantile function at.

```
R> predict(mtmbo, type = "quantile", prob = 0.2, newdata = Prostate[1:5,])
```

```
prob     [,1]  [,2]   [,3]   [,4] [,5]
    0.2 0.181 0.123 0.0456 -0.025 1.24
```

Another method offered by this package implements parametric bootstrap-based sampling. In particular, `simulate.tramnet()` calls the `simulate.ctm()` function after converting the `"tramnet"` object to a `"ctm"` object.

```
R> simulate(mtmbo, nsim = 1, newdata = Prostate[1:5,], seed = 1)
```

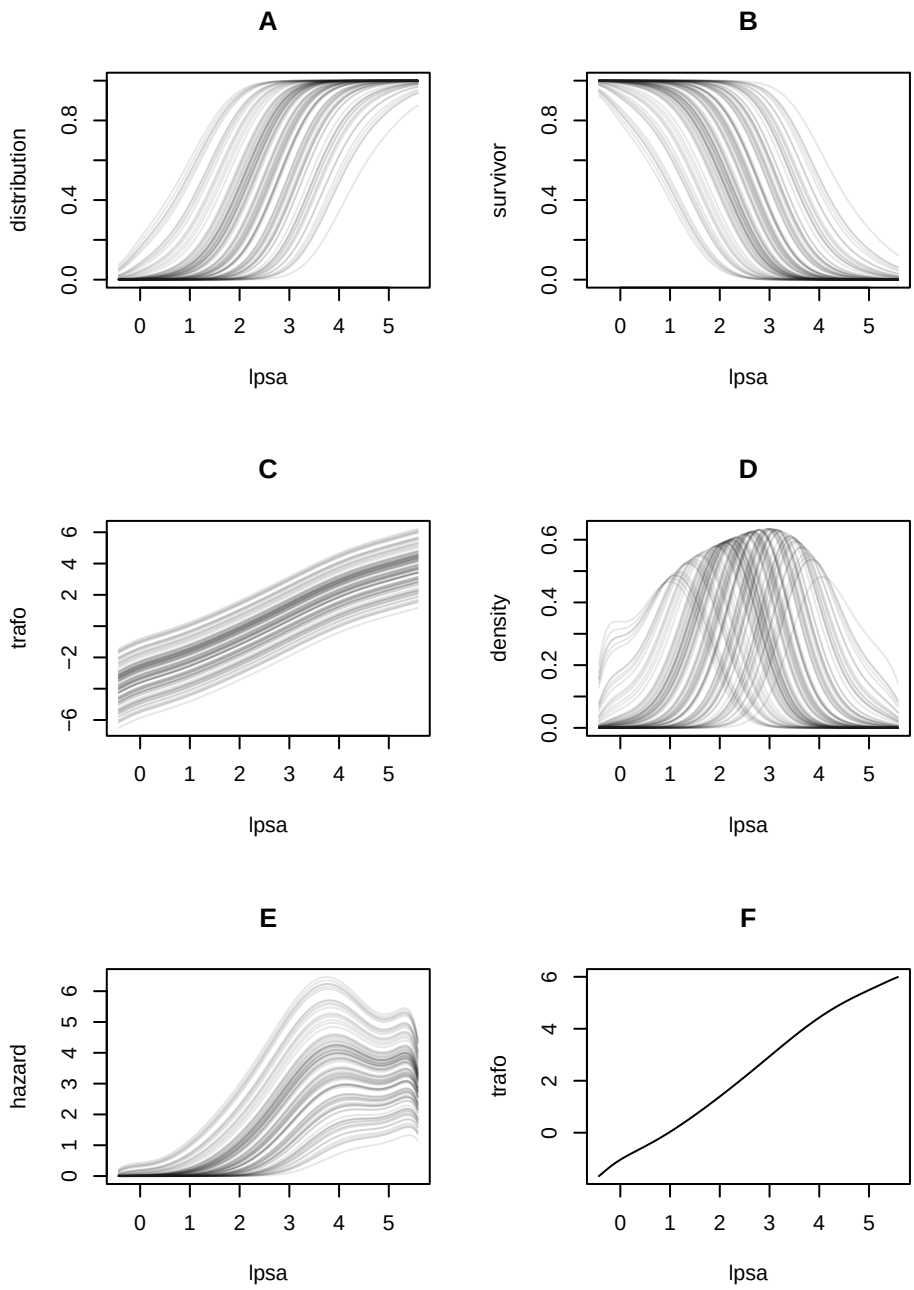


Figure 3: Illustration of `plot.tramnet()`'s versatility in visualizing the response's estimated conditional distribution on various scales including cdf, survivor, transformation scale and pdf. Note that by default the plot is produced for each row in the design matrix. In unstratified linear transformation models this leads to shifted versions of the same curve on the transformation function's scale. A: Estimated conditional distribution function for every observation. B: Estimated conditional survivor function for every observation. The conditional survivor function is defined as $S(y|\boldsymbol{x}) = 1 - F_Y(y|\boldsymbol{x})$. C: Conditional most likely transformation for every observation. Note that every conditional transformation function is a shifted version of the same curve. D: The conditional density for every observation can be calculated using $f_Y(y|\boldsymbol{x}) = F'(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta} - \boldsymbol{x}^\top \boldsymbol{\beta})\boldsymbol{a}'(y)^\top \boldsymbol{\vartheta}$. E: A distribution function is fully characterized by its hazard function $\lambda(y|\boldsymbol{x}) = f_Y(y|\boldsymbol{x})/S(y|\boldsymbol{x})$, which is depicted in this panel. F: The `newdata` argument can be used to plot the predicted most likely transformation for the provided data, in this case the first row of the `Prostate` data.

```
[1] 0.395 0.628 1.004 1.800 1.242
```

Lastly, `residuals.tramnet()` computes the generalized residual r defined as the score contribution for sample i with respect to a newly introduced intercept parameter γ, which is restricted to be zero. In particular,

$$r = \left.\frac{\partial \ell\left(\boldsymbol{\vartheta}, \boldsymbol{\beta}, \gamma; y, \boldsymbol{s}, \boldsymbol{x}\right)}{\partial \gamma}\right|_{\gamma=0}$$

yields the generalized residual with respect to γ for the model

$$F_Y(y|\boldsymbol{s}, \boldsymbol{x}) = F\left(h(y \mid \boldsymbol{s}) - \boldsymbol{x}^\top \boldsymbol{\beta} - \gamma\right).$$

```
R> residuals(mtmbo)[1:5]
[1] -1.67 -1.19 -1.10 -1.02 -1.82
```

In residual analysis and boosting it is common practice to check for associations between residuals and covariates that are not included in the model. In the prostate cancer example one could investigate, whether the variables `age` and `lcp` should be included in the model. To illustrate this particular case a non-parametric `independence_test()` from package `coin` can be used (Hothorn et al., 2008). First, the uncoditional transformation model `m0` is fit. Afterwards, the tramnet models excluding `age` and `lcp` are estimated and their residuals extracted using the `residuals.tramnet()` method. Lastly, an independence test using a maximum statistic (`teststat = "max"`) and a Monte Carlo based approximation of the null distribution based on resampling 10^6 times (`distribution = approximate(1e6)`), yields the results printed below.

```
R> library("coin")
R> m0 <- BoxCox(lpsa ~ 1, data = Prostate, extrapolate = TRUE)
R> x_no_age_lcp <- x[, !colnames(x) %in% c("age", "lcp")]
R> mt_no_age_lcp <- tramnet(m0, x_no_age_lcp, alpha = 0, lambda = 0)
R> r <- residuals(mt_no_age_lcp)
R> it <- independence_test(r ~ age + lcp, data = Prostate,
+                           teststat = "max", distribution = approximate(1e6))
R> pvalue(it, "single-step")

age 0.308
lcp 0.766


age  0.024228
lcp <0.000001
```

Because there is substantial evidence against independence of the models' residuals to either lcp or age, we can conclude that it is worthwhile to include `age` and `lcp` in the model. Packages `trtf` (Hothorn, 2022) and `tbm` (Hothorn, 2020a, 2021) make use of this definition of a residual for estimating and boosting transformation models, trees and random forests. For more theoretical insight the reader is referred to the above mentioned publications.

References

Bernd Bischl, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. *mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions*, 2017. URL https://arxiv.org/abs/1703.03373.

George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 26(2):211–243, 1964. doi: 10.1111/j.2517-6161.1964.tb00553.x.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004. doi: 10.1214/009053604000000067.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. doi: 10.18637/jss.v033.i01.

Anqi Fu, Balasubramanian Narasimhan, and Stephen Boyd. CVXR: An R package for disciplined convex optimization. *Journal of Statistical Software*, 2020. doi: https://arxiv.org/abs/1711.07582. Accepted for publication.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

J. J. Goeman. L1 penalized estimation in the Cox proportional hazards model. *Biometrical Journal*, 52(1):–14, 2010. doi: 10.1002/bimj.200900028.

Michael Grant, Stephen Boyd, and Yinyu Ye. Disciplined convex programming. In *Global Optimization*, pages 155–210. Springer, 2006. doi: 10.1007/s11590-019-01422-z.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634.

Daniel Horn and Bernd Bischl. Multi-objective parameter configuration of machine learning algorithms using model-based optimization. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016. doi: 10.1109/SSCI.2016.7850221.

Torsten Hothorn. Transformation boosting machines. *Statistics and Computing*, 30:141–152, 2020a. doi: 10.1007/s11222-019-09870-4.

Torsten Hothorn. Most likely transformations: The mlt package. *Journal of Statistical Software*, 92(1):1–68, 2020b. doi: 10.18637/jss.v092.i01.

Torsten Hothorn. *tbm: Transformation Boosting Machines*, 2021. URL https://CRAN.R-project.org/package=tbm. R package version 0.3-4.

Torsten Hothorn. *trtf: Transformation Trees and Forests*, 2022. URL https://CRAN.R-project.org/package=trtf. R package version 0.4-0.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, and Achim Zeileis. Implementing a class of permutation tests: The coin package. *Journal of Statistical Software*, 28(8):1–23, 2008. doi: 10.18637/jss.v028.i08.

Torsten Hothorn, Thomas Kneib, and Peter Bühlmann. Conditional transformation models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1):3–27, 2014. doi: 10.1111/rssb.12017.

Torsten Hothorn, Lisa Möst, and Peter Bühlmann. Most likely transformations. *Scandinavian Journal of Statistics*, 45(1):110–134, 2018. doi: 10.1111/sjos.12291.

Torsten Hothorn, Luisa Barbanti, and Sandra Siegfried. *tram: Transformation Models*, 2023. URL https://CRAN.R-project.org/package=tram. R package version 0.8-1.

Lucas Kook and Torsten Hothorn. Regularized transformation models: The tramnet package. *The R Journal*, 13(1):581–594, 2021. doi: 10.32614/RJ-2021-054.

T Lohse, S Rohrmann, D Faeh, and T Hothorn. Continuous outcome logistic regression for analyzing body mass index distributions. *F1000Research*, 6(1933), 2017. doi: 10.12688/f1000research.12934.1.

Thomas A Stamey, John N Kabalin, John E McNeal, Iain M Johnstone, Fuad Freiha, Elise A Redwine, and Norman Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. ii. radical prostatectomy treated patients. *The Journal of Urology*, 141(5):1076–1083, 1989. doi: 10.1016/S0022-5347(17)41176-1.

Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 58(1):267–288, 1996. doi: 10.1111/j.2517-6161.1996.tb02080.x.

Andrey Nikolayevich Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943. doi: 10.1155/2011/450269.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. doi: 10.1111/j.1467-9868.2005.00503.x.

```
R version 4.6.1 (2026-06-24)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 26.04.1 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.32.so;  LAPACK version

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] colorspace_2.1-4 lattice_0.23-1   kableExtra_1.4.1 coin_1.4-6
 [5] glmnet_5.1       Matrix_1.7-6     penalized_0.9-53 survival_3.8-12
 [9] tramnet_0.1-0    CVXR_1.9.2       tram_1.4-7       mvtnorm_1.4-2
[13] mlt_1.8-4        basefun_1.2-7    variables_1.1-2

loaded via a namespace (and not attached):
 [1] viridisLite_0.4.3    orthopolynom_1.0-6.1 farver_2.1.2
 [4] libcoin_1.0-13       S7_0.2.2             fastmap_1.2.0
 [7] TH.data_1.1-5        icenReg_2.0.16       paradox_1.0.1
[10] digest_0.6.39        lifecycle_1.0.5      magrittr_2.0.5
[13] compiler_4.6.1       rlang_1.3.0          tools_4.6.1
[16] data.table_1.18.6.1  knitr_1.52           BB_2026.1.0
[19] spacefillr_0.4.0     alabama_2025.1.0     xml2_1.6.0
[22] RColorBrewer_1.1-3   multcomp_1.4-32      numDeriv_2016.8-1.1
[25] mlr3misc_0.23.0      sys_3.4.3            grid_4.6.1
[28] clarabel_0.11.3      stats4_4.6.1         future_1.80.0
[31] scales_1.4.0         globals_0.20.0       iterators_1.0.14
[34] MASS_7.3-66          cli_3.6.6            rmarkdown_2.32
[37] crayon_1.5.3         bbotk_1.13.1         otel_0.2.0
[40] rstudioapi_0.19.0    scs_3.2.7            polynom_1.4-1
[43] stringr_1.6.0        modeltools_0.2-25    splines_4.6.1
[46] parallel_4.6.1       vctrs_0.7.3          matrixStats_1.5.0
[49] sandwich_3.1-3       mlr3tuning_1.7.0     Formula_1.2-6
[52] listenv_1.1.0        systemfonts_1.3.2    mlr3learners_0.16.0
[55] maketools_1.3.2      foreach_1.5.2        lgr_0.5.2
[58] glue_1.8.1           parallelly_1.48.0    nloptr_2.2.1
[61] codetools_0.2-20     stringi_1.8.9        shape_1.4.6.1
[64] mlr3_1.8.0           palmerpenguins_0.1.1 quadprog_1.5-8
```

```
[67] osqp_1.0.0          gmp_0.7-5.1          coneproj_1.23
[70] htmltools_0.5.9     R6_2.6.1             textshaping_1.0.5
[73] lhs_1.3.0           Rdpack_2.6.6         evaluate_1.0.5
[76] moocore_0.3.2       rbibutils_2.4.1      backports_1.5.1
[79] highs_1.14.0-2      mlr3mbo_1.2.1        Rcpp_1.1.2
[82] uuid_1.2-2          svglite_2.2.2        coda_0.19-4.1
[85] checkmate_2.3.4     xfun_0.62            zoo_1.9-1
[88] buildtools_1.0.0
```