Most Likely Transformations: The mlt Package

Torsten Hothorn
Universität Zürich

Abstract

The **mlt** package implements maximum likelihood estimation in the class of conditional transformation models. Based on a suitable explicit parameterisation of the unconditional or conditional transformation function using infrastructure from package **basefun**, we show how one can define, estimate, and compare a cascade of increasingly complex transformation models in the maximum likelihood framework. Models for the unconditional or conditional distribution function of any univariate response variable are set-up and estimated in the same computational framework simply by choosing an appropriate transformation function and parameterisation thereof. As it is computationally cheap to evaluate the distribution function, models can be estimated by maximisation of the exact likelihood, especially in the presence of random censoring or truncation. The relatively dense high-level implementation in the R system for statistical computing allows generalisation of many established implementations of linear transformation models, such as the Cox model or other parametric models for the analysis of survival or ordered categorical data, to the more complex situations illustrated in this paper.

Keywords: transformation model, transformation analysis, distribution regression, conditional distribution function, conditional quantile function, censoring, truncation.

1. Introduction

The history of statistics can be told as a story of great conceptual ideas and contemporaneous computable approximations thereof. As time went by, the computationally inaccessible concept often vanished from the collective consciousness of our profession and the approximation was taught and understood as the real thing. Least squares regression emerged from Gauß' computational trick of changing Bošcović' absolute to squared error and it took 200 years for the original, and in many aspects advantageous, concept to surface again under the name "quantile regression". This most prominent example of an idea got lost illustrates the impact computable approximations had and still have on our understanding of statistical methods and procedures. In the early days of statistical computing, implementations of such approximations were a challenge. With today's computing power and software infrastructure at our fingertips, our duty shall be to go back to the original concepts and search for ways how to reawake them for the benefit of a simpler understanding of statistical models and concepts.

This paper describes an attempt to understand and unify a large class of statistical models as

Please cite this document as: Torsten Hothorn (2017) Most Likely Transformations: The **mlt** Package. R package vignette version 0.2-1, URL https://CRAN.R-project.org/package=mlt.docreg.

models for unconditional or conditional distributions. This sounds like an implicitness, but do we really practice (in courses on applied statistics or while talking to our subject-matter collaborators) what we preach in a theory course? Let's perform a small experiment: Pick, at random, a statistics book from your book shelf and look-up how the general linear model is introduced. Most probably you will find something not unlike

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

where Y is an absolutely continuous response, $\tilde{\boldsymbol{x}}$ a suitable representation of a vector of explanatory variables $\boldsymbol{x}$ (*e.g.* contrasts etc.), $\boldsymbol{\beta}$ a vector of regression coefficients, α an intercept term and ε a normal error term. Model interpretation relies on the conditional expectation $\mathbb{E}(Y \mid \boldsymbol{X} = \boldsymbol{x}) = \alpha + \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}$. Many textbooks, for example Fahrmeir *et al.* (2013), *define* a regression model as a model for a conditional expectation $\mathbb{E}(Y \mid \boldsymbol{X} = \boldsymbol{x}) = f(\boldsymbol{x})$ with "regression function" f but, as we will see, understanding regression models as models for conditional distributions makes it easier to see the connections between many classical and novel regression approaches. In the linear model, the intercept α and the regression parameters $\boldsymbol{\beta}$ are estimated by minimisation of the squared error $(Y - \alpha - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta})^2$. With some touch-up in notation, the model can be equivalently written as a model for a conditional distribution

$$\mathbb{P}(Y \le y \mid \boldsymbol{X} = \boldsymbol{x}) = \Phi\left(\frac{y - \alpha - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}}{\sigma}\right) \text{ or } (Y \mid \boldsymbol{X} = \boldsymbol{x}) \sim \mathrm{N}(\alpha + \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}, \sigma^2).$$

This formulation highlights that the model is, in fact, a model for a conditional distribution and not just a model for a conditional mean. It also stresses the fact that the variance σ^2 is a model parameter in its own right. The usual treatment of σ^2 as a nuisance parameter only works when the likelihood is approximated by the density of the normal distribution. Because in real life we always observe intervals $(\underline{y}, \bar{y}]$ and never real numbers y, the exact likelihood is, as originally *defined* by Fisher (1934)

$$\mathbb{P}(\underline{y} < Y \le \bar{y} \mid \boldsymbol{X} = \boldsymbol{x}) = \Phi\left(\frac{\bar{y} - \alpha - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}}{\sigma}\right) - \Phi\left(\frac{\underline{y} - \alpha - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}}{\sigma}\right)$$

which requires simultaneous optimisation of all three model parameters α, $\boldsymbol{\beta}$ and σ but is exact also under other forms of random censoring. This exact likelihood is another prominent example of its approximation (via the log-density $(y - \alpha - \tilde{\boldsymbol{x}}^\top)^2$) winning over the basic concept, see Lindsey (1996, 1999). If we were going to reformulate the model a little further to

$$\mathbb{P}(Y \le y \mid \boldsymbol{X} = \boldsymbol{x}) = \Phi(\tilde{\alpha}_1 + \tilde{\alpha}_2 y - \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{\beta}})$$

with $\tilde{\alpha}_1 = -\alpha/\sigma, \tilde{\alpha}_2 = 1/\sigma$ and $\tilde{\boldsymbol{\beta}} = \boldsymbol{\beta}/\sigma$ we see that the model is of the form

$$\mathbb{P}(Y \le y \mid \boldsymbol{X} = \boldsymbol{x}) = F_Z(h_Y(y) - \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{\beta}})$$

with distribution function $F_Z = \Phi$ and linear transformation $h_Y(y) = \tilde{\alpha}_1 + \tilde{\alpha}_2 y$ such that $\mathbb{E}(h_Y(Y) \mid \boldsymbol{X} = \boldsymbol{x}) = \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}$. If we now change F_Z to the distribution function of the minimum extreme value distribution and allow a non-linear monotone transformation h_Y we get

$$\mathbb{P}(Y \le y \mid \boldsymbol{X} = \boldsymbol{x}) = 1 - \exp(-\exp(h_Y(y) - \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{\beta}}))$$

which is the continuous proportional hazards, or Cox, model (typically defined with a positive shift term). From this point of view, the linear and the Cox model are two instances of so-called linear transformation models (a misleading name, because the transformation h_Y of the response Y is non-linear in the latter case and only the shift term $\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{\beta}}$ is linear in $\tilde{\boldsymbol{x}}$). It is now also obvious that the Cox model has nothing to do with censoring, let alone survival times $Y > 0$. It is a model for the conditional distribution of a continuous response $Y \in \mathbb{R}$ when it is appropriate to assume that the conditional hazard function is scaled by $\exp(\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{\beta}})$. For both the linear and the Cox model, application of the exact likelihood allows the models to be fitted to imprecise, or "censored", observations $(\underline{y}, \bar{y}] \in \mathbb{R}$. The generality of the class of linear transformation models comes from the ability to change F_Z and h_Y in this flexible class of models.

The class of linear transformation models is a subclass of conditional transformation models (Hothorn *et al.* 2014). In this latter class, the conditional distribution function is modelled by

$$\mathbb{P}(Y \le y \mid \boldsymbol{X} = \boldsymbol{x}) = F_Z(h(y \mid \boldsymbol{x}))$$

where the transformation function h depends on both y and $\boldsymbol{x}$. This function is unknown and this document describes how one can estimate h. Because we are interested in analysing, *i.e.* estimating and interpreting, the transformation function h, we slightly rephrase the title "An Analysis of Transformations" of the first paper on this subject (Box and Cox 1964) and refer to the methods presented in this paper as *transformation analysis*. We describe how conditional transformation models can be specified, fitted by maximising the likelihood (using the estimator developed by Hothorn *et al.* 2017b) and analysed in R using the **mlt** add-on package (Hothorn 2017b). In essence, the package is built around two important functions. `ctm()` specifies transformation models based on basis functions implemented in the **basefun** package (Hothorn 2017a) for variable descriptions from the **variables** package (Hothorn 2016). These models are then fitted to data by `mlt()`. This function maximises the exact or approximate log-likelihood function using the `auglag()` and `spg()` optimisers from packages **alabama** (Varadhan 2015) and **BB** (Varadhan and Gilbert 2009, 2014), respectively. Before we start looking at details, we illustrate the workflow by means of an example from unconditional density estimation.

Density Estimation: Old Faithful Geyser Data The duration of eruptions and the waiting time between eruptions of the Old Faithful geyser in the Yellowstone national park became a standard benchmark for non-parametric density estimation (the original data were given by Azzalini and Bowman 1990). An unconditional density estimate for the duration of the eruptions needs to deal with censoring because exact duration times are only available for the day time measurements. At night time, the observations were either left-censored ("short" eruption), interval-censored ("medium" eruption) or right-censored ("long" eruption) as explained by Azzalini and Bowman (1990). This fact was widely ignored in analyses of the Old Faithful data because most non-parametric density estimators cannot deal with censoring. The key issue is the representation of the unknown transformation function by $h(y) = \boldsymbol{a}(y)^\top \boldsymbol{\vartheta}$ based on suitable basis functions $\boldsymbol{a}$ and parameters $\boldsymbol{\vartheta}$. We fit these parameters $\boldsymbol{\vartheta}$ in the transformation model

$$\mathbb{P}(Y \le y) = \Phi(h(y)) = \Phi(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta})$$

by maximisation of the exact likelihood as follows. After loading package **mlt** we specify the `duration` variable we are interested in

```
R> library("mlt")
R> var_d <- numeric_var("duration", support = c(1.0, 5.0),
+                        add = c(-1, 1), bounds = c(0, Inf))
```

This abstract representation refers to a positive and conceptually continuous variable `duration`. We then set-up a basis function $\boldsymbol{a}$ for this variable in the interval $[1, 5]$ (which can be evaluated in the interval $[0, 6]$ as defined by the `add` argument), in our case a monotone increasing Bernstein polynomial of order eight (details can be found in Section 2.1)

```
R> B_d <- Bernstein_basis(var = var_d, order = 8, ui = "increasing")
```

The (in our case unconditional) transformation model is now fully defined by the parameterisation $h(y) = \boldsymbol{a}(y)^\top \boldsymbol{\vartheta}$ and $F_Z = \Phi$ which is specified using the `ctm()` function as

```
R> ctm_d <- ctm(response = B_d, todistr = "Normal")
```

Because, in this simple case, the transformation function transforms $Y \sim F_Y$ to $Z \sim F_Z = \Phi$, the latter distribution is specified using the `todistr` argument. An equidistant grid of 200 duration times in the interval `support + add` $= [0, 6]$ is generated by

```
R> str(nd_d <- mkgrid(ctm_d, 200))

List of 1
 $ duration: num [1:200] 0 0.0302 0.0603 0.0905 0.1206 ...
```

Note that the model `ctm_d` has no notion of the actual observations. The `support` argument of `numeric_var` defines the domain of the Bernstein polynomial and is the only reference to the data. Only after the model was specified we need to load the data frame containing the observations of `duration` as a `Surv` object

```
R> data("geyser", package = "TH.data")
R> head(geyser)

  waiting         duration
1      80         4.016667
2      71         2.150000
3      57 [4.000000, Inf]
4      80 [4.000000, Inf]
5      75 [4.000000, Inf]
6      77 [0.000000,   2]
```

The most likely transformation (MLT) $\hat{h}_N(y) = \boldsymbol{a}(y)^\top \hat{\boldsymbol{\vartheta}}_N$ is now obtained from the maximum likelihood estimate $\hat{\boldsymbol{\vartheta}}_N$ computed as

```
R> mlt_d <- mlt(ctm_d, data = geyser)
R> logLik(mlt_d)
```

```
'log Lik.' -317.766 (df=9)
```

The model is best visualised in terms of the corresponding density $\phi(\boldsymbol{a}(y)^\top \hat{\boldsymbol{\vartheta}}_N)\boldsymbol{a}'(y)^\top \hat{\boldsymbol{\vartheta}}_N$, which can be extracted from the fitted model using

```
R> nd_d$d <- predict(mlt_d, newdata = nd_d, type = "density")
```

The estimated density is depicted in Figure 1. The plot shows the well-known bimodal distribution in a nice smooth way. Several things are quite unusual in this short example. First, the model was specified using `ctm()` without reference to the actual observations, and second, although the model is fully parametric, the resulting density resembles the flexibility of a non-parametric density estimate (details in Section 2). Third, the exact likelihood, as defined by the interval-censored observations, was used to obtain the model (Section 3). Fourth, inspection of the parameter estimates $\hat{\boldsymbol{\vartheta}}_N$ is uninteresting, the model is better looked at by means of the estimated distribution, density, quantile, hazard or cumulative hazard functions (Section 4). Fifth, no regularisation is necessary due to the monotonicity constraint on the estimated transformation function (implemented as linear constraints for maximum likelihood estimation) and thus standard likelihood asymptotics work for $\hat{\boldsymbol{\vartheta}}_N$ (Section 5). Sixth, because the model is a model for a full distribution, we can easily draw random samples from the model and refit its parameters using the parametric or model-based bootstrap (Section 6). Seventh, all of this is not only possible theoretically but readily implemented in package **mlt**. The only remaining question is "Do all these nice properties carry over to the conditional case, *i.e.* to regression models?". The answer to this question is "yes!" and the rest of this paper describes the details following the workflow sketched in this section. Technical issues about the three packages **variables**, **basefun** and **mlt** involved are explained in Appendices A, B, and C.

2. Specifying Transformation Models

In this section we review a cascade of increasingly complex transformation models following Hothorn *et al.* (2017b) and discuss how one can specify such models using the `ctm()` function provided by the **mlt** package. The models are fitted via maximum likelihood by the `mlt()` function (details are given in Section 3) to studies from different domains. Results obtained from established implementations of the corresponding models in the R universe are used to evaluate the implementation in package **mlt** whereever possible. We start with the simplest case of models for unconditional distribution functions.

2.1. Unconditional Transformation Models

The distribution of an at least ordered response $Y \in \Xi$ is defined by a transformation function h and a distribution function F_Z. The transformation function is parameterised in terms of a basis function $\boldsymbol{a}$

$$\mathbb{P}(Y \le y) = F_Z(h(y)) = F_Z(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta}).$$

The triple $(F_Z, \boldsymbol{a}, \boldsymbol{\vartheta})$ fully defines the distribution of Y and is called *transformation model.* The choice of the basis function $\boldsymbol{a}$ depends on the measurement scale of Y and we can differentiate between the following situations.

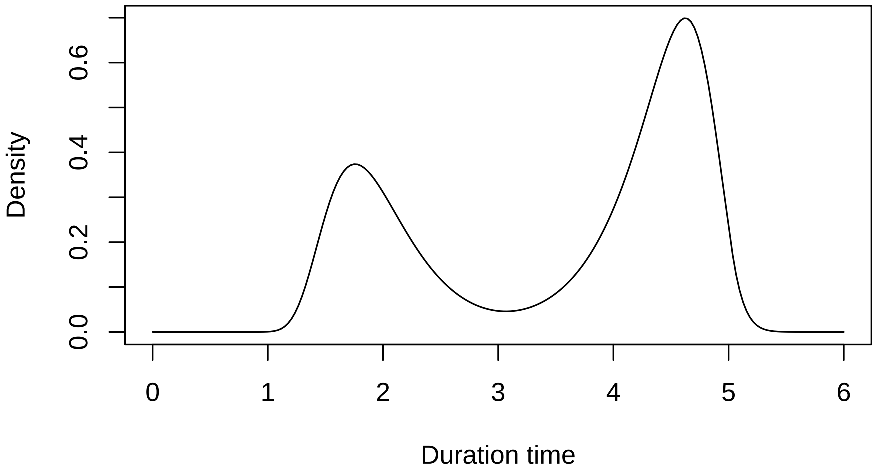

Figure 1: Old Faithful Geyser. Estimated density for duration time obtained from an unconditional transformation model where the transformation function was parameterised as a Bernstein polynomial of order eight. The plot reproduces Figure 1 (right panel) in Hothorn *et al.* (2017c).

Discrete Models for Categorical Responses

For ordered categorical responses Y from a finite sample space $\Xi = \{y_1, \dots, y_K\}$ the distribution function F_Y is a step-function with jumps at y_k only. We therefore assign one parameter to each jump, *i.e.* each element of the sample space except y_K. This corresponds to the basis function $\boldsymbol{a}(y_k) = \boldsymbol{e}_{K-1}(k)$, where $\boldsymbol{e}_{K-1}(k)$ is the unit vector of length $K-1$ with its kth element being one. The transformation function h is

$$h(y_k) \quad = \quad \boldsymbol{e}_{K-1}(k)^\top \boldsymbol{\vartheta} = \vartheta_k \in \mathbb{R}, \quad 1 \le k < K, \quad \text{st} \quad \vartheta_1 < \dots < \vartheta_{K-1}$$

with $h(y_K) = \infty$, and the unconditional distribution function of F_Y is $F_Y(y_k) = F_Z(\vartheta_k)$. Note that monotonicity of h is guaranteed by the $K-2$ linear constraints $\vartheta_2 - \vartheta_1 > 0, \dots, \vartheta_{K-1} - \vartheta_{K-2} > 0$ when constrained optimisation is performed. The density of a nominal variable Y can be estimated in the very same way because it is invariant with respect to the ordering of the levels (see Section 3).

Categorical Data Analysis: Chinese Health and Family Life Survey The Chinese Health and Family Life Survey (Parish and Laumann 2009), conducted 1999–2000 as a collaborative research project of the Universities of Chicago, Beijing, and North Carolina, sampled 60 villages and urban neighbourhoods in China. Eighty-three individuals were chosen at random for each location from official registers of adults aged between 20 and 64 years to target a sample of 5000 individuals in total. Here, we restrict our attention to women with current male partners for whom no information was missing, leading to a sample of 1534 women with the following variables: R_edu (level of education of the responding woman), R_income (monthly income in Yuan of the responding woman), R_health (health status of the responding woman in the last year) and R_happy (how happy was the responding woman in the last year). We first estimate the unconditional distribution of happiness using a proportional odds model (polr() from package **MASS**, Ripley 2017)

```
R> data("CHFLS", package = "HSAUR3")
R> polr_CHFLS_1 <- polr(R_happy ~ 1, data = CHFLS)
```

The basis function introduced above corresponds to a model matrix for the ordered factor R_happy with treatment contrasts using the largest level ("very happy") as baseline group. In addition, the parameters must satisfy the linear constraint $\boldsymbol{C}\boldsymbol{\vartheta} \geq \boldsymbol{m}$, $\boldsymbol{C}$ (argument ui) being the difference matrix and $\boldsymbol{m} = \boldsymbol{0}$ (argument ci)

```
R> nl <- nlevels(CHFLS$R_happy)
R> b_happy <- as.basis(~ R_happy, data = CHFLS, remove_intercept = TRUE,
+                      contrasts.arg = list(R_happy = function(n)
+                          contr.treatment(n, base = nl)),
+                      ui = diff(diag(nl - 1)), ci = rep(0, nl - 2))
```

A short-cut for ordered factors avoids this rather complex definition of the basis function

```
R> b_happy <- as.basis(CHFLS$R_happy)
```

We are now ready to set-up the (unconditional) transformation model by a call to ctm() using the basis function and a character defining the standard logistic distribution function for F_Z

```
R> ctm_CHFLS_1 <- ctm(response = b_happy, todist = "Logistic")
```

Note that the choice of F_Z is completely arbitrary as the estimated distribution function is invariant with respect to F_Z. The model is fitted by calling the mlt() function with arguments model and data

```
R> mlt_CHFLS_1 <- mlt(model = ctm_CHFLS_1, data = CHFLS)
```

The results are equivalent to the results obtained from polr()

```
R> logLik(polr_CHFLS_1)

'log Lik.' -1328.241 (df=3)

R> logLik(mlt_CHFLS_1)

'log Lik.' -1328.241 (df=3)

R> cbind(polr = polr_CHFLS_1$zeta, mlt = coef(mlt_CHFLS_1))
```

```
                                      polr       mlt
Very unhappy|Not too happy   -4.687408 -4.687544
Not too happy|Somewhat happy -1.903382 -1.903378
Somewhat happy|Very happy     1.499304  1.499261
```

Of course, the above exercise is an extremely cumbersome way of estimating a discrete density whose maximum likelihood estimator is a simple proportion but, as we will see in the rest of this paper, a generalisation to the conditional case strongly relies on this parameterisation

```
R> cbind(polr = predict(polr_CHFLS_1, newdata = data.frame(1), type = "prob"),
+        mlt = c(predict(mlt_CHFLS_1, newdata = data.frame(1),
+                        type = "density", q = mkgrid(b_happy)[[1]])),
+        direct = xtabs(~ R_happy, data = CHFLS) / nrow(CHFLS))
```

```
                      polr        mlt      direct
Very unhappy   0.009126466 0.00912524 0.009126467
Not too happy  0.120599751 0.12060135 0.120599739
Somewhat happy 0.687744449 0.68773768 0.687744459
Very happy     0.182529334 0.18253574 0.182529335
```

Continuous Models for Continuous Responses

For continuous responses $Y \in \Xi \subseteq \mathbb{R}$ the parameterisation $h(y) = \boldsymbol{a}(y)^\top \boldsymbol{\vartheta}$ should be smooth in y, so any polynomial or spline basis is a suitable choice for $\boldsymbol{a}$. We apply Bernstein polynomials (for an overview see Farouki 2012) of order M (with $M+1$ parameters) defined on the interval $[\underline{\imath}, \bar{\imath}]$ with

$$
\begin{aligned}
\boldsymbol{a}_{\text{Bs},M}(y) &= (M+1)^{-1}(f_{\text{Be}(1,M+1)}(\tilde{y}), \dots, f_{\text{Be}(m,M-m+1)}(\tilde{y}), \dots, f_{\text{Be}(M+1,1)}(\tilde{y}))^\top \in \mathbb{R}^{M+1} \\
h(y) &= \boldsymbol{a}_{\text{Bs},M}(y)^\top \boldsymbol{\vartheta} = \sum_{m=0}^{M} \vartheta_m f_{\text{Be}(m+1,M-m+1)}(\tilde{y})/(M+1) \\
h'(y) &= \boldsymbol{a}'_{\text{Bs},M}(y)^\top \boldsymbol{\vartheta} = \sum_{m=0}^{M-1} (\vartheta_{m+1} - \vartheta_m) f_{\text{Be}(m+1,M-m)}(\tilde{y}) M/((M+1)(\bar{\imath} - \underline{\imath}))
\end{aligned}
$$

where $\tilde{y} = (y - \underline{\imath})/(\bar{\imath} - \underline{\imath}) \in [0,1]$ and $f_{\text{Be}(m,M)}$ is the density of the Beta distribution with parameters m and M. This choice is computationally attractive because strict monotonicity can be formulated as a set of M linear constraints on the parameters $\vartheta_m < \vartheta_{m+1}$ for all $m = 0, \dots, M$ (Curtis and Ghosh 2011). Therefore, application of constrained optimisation guarantees monotone estimates $\hat{h}_N$. The basis contains an intercept.

Density Estimation: Geyser Data (Cont'd) We continue the analysis of the Old Faithful data by estimating the unconditional distribution of waiting times, a positive variable whose abstract representation can be used to generate an equidistant grid of 100 values

```
R> var_w <- numeric_var("waiting", support = c(40.0, 100), add = c(-5, 15),
+                       bounds = c(0, Inf))
R> c(sapply(nd_w <- mkgrid(var_w, 100), range))
```

```
[1]  35 115
```

A monotone increasing Bernstein polynomial of order eight for `waiting` in the interval $[\underline{\imath}, \bar{\imath}]$ `= support(var_w)` is defined as

```
R> B_w <- Bernstein_basis(var_w, order = 8, ui = "increasing")
```

The (here again unconditional) transformation model is now fully described by the parameterisation $h(y) = \boldsymbol{a}(y)^\top \boldsymbol{\vartheta}$ and $F_Z = \Phi$, the latter choice again being not important (for larger values of M)

```
R> ctm_w <- ctm(response = B_w, todistr = "Normal")
```

The most likely transformation $\hat{h}_N(y) = \boldsymbol{a}(y)^\top \hat{\boldsymbol{\vartheta}}_N$ is now obtained from the maximum likelihood estimate $\hat{\boldsymbol{\vartheta}}_N$ computed as

```
R> mlt_w <- mlt(ctm_w, data = geyser)
```

and we compare the estimated distribution function

```
R> nd_w$d <- predict(mlt_w, newdata = nd_w, type = "distribution")
```

with the empirical cumulative distribution function (the non-parametric maximum likelihood estimator) in the left panel of Figure 2.

The question arises how the degree of the polynomial affects the estimated distribution function. On the one hand, the model $(\Phi, \boldsymbol{a}_{\text{Bs},1}, \boldsymbol{\vartheta})$ only allows linear transformation functions of a standard normal and F_Y is restricted to the normal family. On the other hand, $(\Phi, \boldsymbol{a}_{\text{Bs},N-1}, \boldsymbol{\vartheta})$ has one parameter for each observation and $\hat{F}_{Y,N}$ is the non-parametric maximum likelihood estimator ECDF which, by the Glivenko-Cantelli lemma, converges to F_Y. In this sense, we cannot choose M "too large". This is a consequence of the monotonicity constraint on the estimator $\boldsymbol{a}^\top \hat{\boldsymbol{\vartheta}}_N$ which, in this extreme case, just interpolates the step-function $F_Z^{-1} \circ \text{ECDF}$. The practical effect can be inspected in Figure 2 where two Bernstein polynomials of order $M = 8$ and $M = 40$ are compared on the scale of the distribution function (left panel) and density function (right panel). It is hardly possible to notice the difference in probabilities but the more flexible model features a more erratic density estimate, but not overly so.

Continuous Models for Discrete Responses

Although a model for Y can assume an absolutely continuous distribution, observations from such a model will always be discrete. This fact is taken into account by the exact likelihood (Section 3). In some cases, for example for count data with potentially large number of counts, one might use a continuous parameterisation of the transformation function $\boldsymbol{a}_{\text{Bs},M}(y)^\top \boldsymbol{\vartheta}$ which is evaluated at the observed counts only as in the next example.

Analysis of Count Data: Deer-vehicle Collisions Hothorn *et al.* (2015) analyse roe deer-vehicle collisions reported to the police in Bavaria, Germany, between 2002-01-01 and 2011-12-31. The daily counts range between 135 and 693. A model for the daily number of roe deer-vehicle collision using a Bernstein polynomial of order six as basis function is fitted using

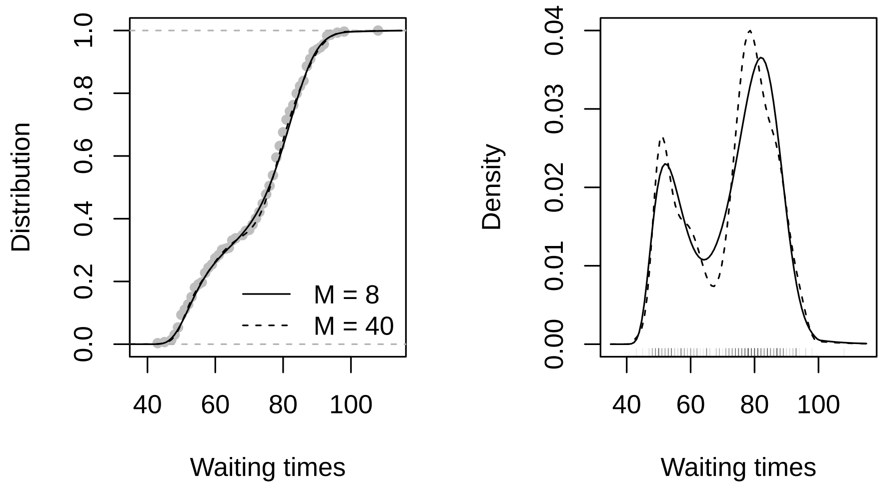

Figure 2: Old Faithful Geyser. Estimated distribution (left, also featuring the empirical cumulative distribution function as grey circles) and density (right) function for two transformation models parameterised in terms of Bernstein polynomials of order eight and 40. The right plot reproduces the density for $M = 8$ shown in Figure 1 (left panel) in Hothorn *et al.* (2017c).

```
R> var_dvc <- numeric_var("dvc", support = min(dvc):max(dvc))
R> B_dvc <- Bernstein_basis(var_dvc, order = 6, ui = "increasing")
R> dvc_mlt <- mlt(ctm(B_dvc), data = data.frame(dvc = dvc))
```

The discrete unconditional distribution function (evaluated for all integers between 135 and 693)

```
R> q <- support(var_dvc)[[1]]
R> p <- predict(dvc_mlt, newdata = data.frame(1), q = q,
+               type = "distribution")
```

along with the unconditional distribution function of the Poisson distribution with rate estimated from the data are given in Figure 3. The empirical cumulative distribution function is smoothly approximated using only seven parameters. There is clear evidence for overdispersion as the variance of the Poisson model is much smaller than the variance estimated by the transformation model.

Discrete Models for Continuous Responses

In some applications one might not be interested in estimating the whole distribution function F_Y for a conceptually absolutely continuous response Y but in probabilities $F_Y(y_k)$ for some grid $y_1, \dots, y_K$ only. The discrete basis function $h(y_k) = \boldsymbol{e}_{K-1}(k)^\top \boldsymbol{\vartheta} = \vartheta_k$ can be used in

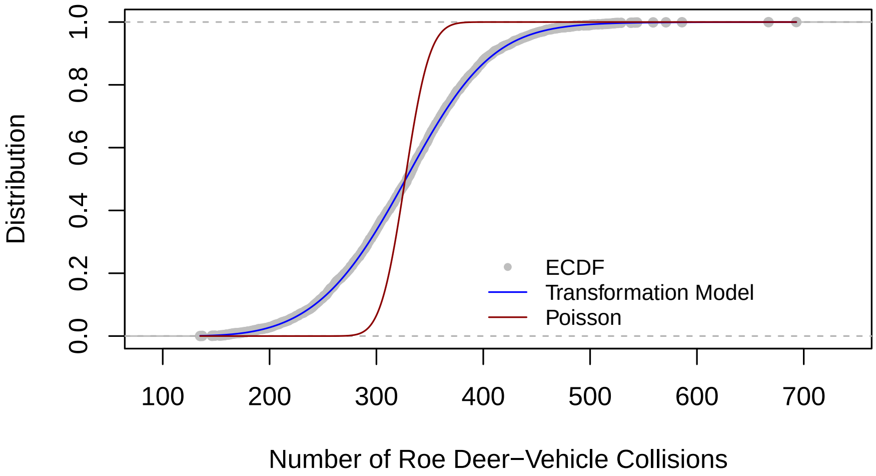

Figure 3: Deer-vehicle Collisions. Unconditional distribution of daily number of roe deer-vehicle collisions in Bavaria, Germany, between 2002 and 2011. Grey circles represent the empirical cumulative distribution function, the blue line the unconditional transformation function and the red line the Poisson distribution fitted to the data.

this case. For model estimation, only the discretised observations $y \in (y_{k-1}, y_k]$ enter the likelihood.

2.2. Linear Transformation Models

A linear transformation model features a linear shift of a typically non-linear transformation of the response Y and is the simplest form of a regression model in the class of conditional transformation models. The conditional distribution function is modelled by

$$\mathbb{P}(Y \le y \mid \boldsymbol{X} = \boldsymbol{x}) = F_Z(h(y \mid \boldsymbol{x})) = F_Z(h_Y(y) - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}).$$

The transformation h_Y, sometimes called “baseline transformation”, can be parameterised as discussed in the unconditional situation using an appropriate basis function $\boldsymbol{a}$

$$F_Z(h_Y(y) - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}) = F_Z(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta}_1 - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}).$$

The connection to more complex models is highlighted by the introduction of a basis function $\boldsymbol{c}$ being conditional on the explanatory variables $\boldsymbol{x}$, *i.e.*

$$F_Z(h(y \mid \boldsymbol{x})) = F_Z(h_Y(y) - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}) = F_Z(\boldsymbol{c}(y, \boldsymbol{x})^\top \boldsymbol{\vartheta}) = F_Z(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta}_1 - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta})$$

with $\boldsymbol{c} = (\boldsymbol{a}^\top, -\tilde{\boldsymbol{x}}^\top)^\top$. The definition of a linear transformation model only requires the basis $\boldsymbol{a}$, the explanatory variables $\boldsymbol{x}$ and a distribution F_Z. The latter choice is now important, as additivity of the linear predictor is assumed on the scale of F_Z. It is convenient to supply negative linear predictors as $\mathbb{E}(h_Y(Y) \mid \boldsymbol{X} = \boldsymbol{x}) = \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}$ (up to an additive constant $\mathbb{E}(Z)$). One exception is the Cox model with positive shift and $\mathbb{E}(h_Y(Y) \mid \boldsymbol{X} = \boldsymbol{x}) = -\tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}$. Large positive values of the linear predictor correspond to low expected survival times and thus a high risk.

Discrete Responses

Categorical Data Analysis: Chinese Survey (Cont'd) We want to study the impact of age and income on happiness in a proportional odds model, here fitted using `polr()` first

```
R> polr_CHFLS_2 <- polr(R_happy ~ R_age + R_income, data = CHFLS)
```

In order to fit this model using the **mlt** package, we need to set-up the basis function for a negative linear predictor without intercept in addition to the basis function for happiness (`b_happy`)

```
R> b_R <- as.basis(~ R_age + R_income, data = CHFLS, remove_intercept = TRUE,
+                  negative = TRUE)
```

The model is now defined by two basis functions, one of the response and one for the shift in addition to F_Z and fitted using the `mlt()` function; the columns of the model matrix are scaled to $[-1, 1]$ before fitting the parameters by `scale = TRUE`

```
R> ctm_CHFLS_2 <- ctm(response = b_happy, shifting = b_R,
+                     todistr = "Logistic")
R> mlt_CHFLS_2 <- mlt(ctm_CHFLS_2, data = CHFLS, scale = TRUE)
```

Again, the results of `polr()` and `mlt()` are equivalent

```
R> logLik(polr_CHFLS_2)

'log Lik.' -1322.021 (df=5)

R> logLik(mlt_CHFLS_2)

'log Lik.' -1322.021 (df=5)

R> cbind(polr = c(polr_CHFLS_2$zeta, coef(polr_CHFLS_2)),
+        mlt = coef(mlt_CHFLS_2))

                                     polr           mlt
Very unhappy|Not too happy   -4.801592360 -4.8016007890
Not too happy|Somewhat happy -2.011695585 -2.0117558752
Somewhat happy|Very happy     1.412707129  1.4126490150
R_age                        -0.006278962 -0.0062807722
R_income                      0.000235009  0.0002350172
```

The regression coefficients $\boldsymbol{\beta}$ are the log-odds ratios, *i.e.* the odds-ratio between any two subsequent happiness categories is 0.9937 for each year of age and 1.0002 for each additional Yuan earned. Therefore, there seems to be a happiness conflict between getting older *and* richer.

Continuous Responses

Survival Analysis: German Breast Cancer Study Group-2 (GBSG-2) Trial This prospective, controlled clinical trial on the treatment of node positive breast cancer patients was conducted by the German Breast Cancer Study Group (GBSG-2, Schumacher *et al.* 1994). Patients not older than 65 years with positive regional lymph nodes but no distant metastases were included in the study. Out of 686 women, 246 received hormonal therapy whereas the control group of 440 women did not receive hormonal therapy. Additional variables include age, menopausal status, tumour size, tumour grade, number of positive lymph nodes, progesterone receptor, and estrogen receptor. The right-censored recurrence-free survival time is the response variable of interest, *i.e.* a positive absolutely continuous variable

```
R> data("GBSG2", package = "TH.data")
R> GBSG2y <- numeric_var("y", support = c(100.0, max(GBSG2$time)),
+                        bounds = c(0, Inf))
R> GBSG2$y <- with(GBSG2, Surv(time, cens))
```

We start with the Cox model $(F_{\text{MEV}}, (\boldsymbol{a}_{\text{Bs},10}^\top, \tilde{\boldsymbol{x}}^\top)^\top, (\boldsymbol{\vartheta}_1^\top, \boldsymbol{\beta}^\top)^\top)$, in more classical notation the model

$$\mathbb{P}(Y \le y \mid \boldsymbol{X} = \boldsymbol{x}) = 1 - \exp(-\exp(\boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_1 + \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}),$$

where $\tilde{\boldsymbol{x}}$ contains the treatment indicator and all other explanatory variables in the transformation function $h(y \mid \boldsymbol{x}) = \boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_1 + \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}$ with log-cumulative baseline hazard $h_Y(y) = \boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_1$ (the positive shift being in line with the implementation in `coxph()` in package **survival**, Therneau and Grambsch 2000; Therneau 2017)

```
R> B_GBSG2y <- Bernstein_basis(var = GBSG2y, order = 10, ui = "increasing")
R> fm_GBSG2 <- Surv(time, cens) ~ horTh + age + menostat + tsize + tgrade +
+                                  pnodes + progrec + estrec
R> ctm_GBSG2_1 <- ctm(B_GBSG2y, shifting = fm_GBSG2[-2L], data = GBSG2,
+                          todistr = "MinExtrVal")
```

`fm_GBSG2[-2L]` is the right hand side of the model formula and defines the basis function for the shift term in the classical formula language. The distribution function F_Z is the distribution function of the minimum extreme value distribution. The specification of the right-hand side of the formula as argument `shifting` along with the data (argument `data`) is equivalent to a basis function

```
R> as.basis(fm_GBSG2[-2L], data = GBSG2, remove_intercept = TRUE)
```

Note that the model matrix is set-up with intercept term first, ensuring proper coding of contrasts. Because the intercept in this model is the log-cumulative baseline hazard function h_Y, the intercept column is removed from the model matrix prior to model estimation.

In contrast to the classical Cox model where only $\boldsymbol{\beta}$ is estimated by the partial likelihood, we estimate all model parameters $(\boldsymbol{\vartheta}_1^\top, \boldsymbol{\beta}^\top)$ simultaneously under ten linear constraints

```
R> mlt_GBSG2_1 <- mlt(ctm_GBSG2_1, data = GBSG2, scale = TRUE)
```

The results obtained for $\boldsymbol{\beta}$ from the partial and the exact log-likelihood are practically equivalent

```
R> coxph_GBSG2_1 <- coxph(fm_GBSG2, data = GBSG2, ties = "breslow")
R> cf <- coef(coxph_GBSG2_1)
R> cbind(coxph = cf, mlt = coef(mlt_GBSG2_1)[names(cf)])
```

```
                     coxph          mlt
horThyes     -0.3462416209 -0.348707936
age          -0.0094534071 -0.009932002
menostatPost  0.2581565452  0.267642672
tsize         0.0077983290  0.007765951
tgrade.L      0.5510838051  0.560223057
tgrade.Q     -0.2011060204 -0.201942423
pnodes        0.0487818043  0.048761983
progrec      -0.0022174942 -0.002210539
estrec        0.0001978178  0.000182990
```

A practically important question is how the order M of the Bernstein polynomial affects the results. Recall that the log-cumulative baseline hazard function h_Y is not specified when the partial likelihood is maximised and thus `coxph()` makes no assumptions regarding this function. To study the impact of M on the results, we refit the Cox model using `mlt()` for $M = 1, \dots, 30$ and plot the log-cumulative hazard function h_Y for different M along with the non-parametric Nelson-Aalen-Breslow estimator in the left panel of Figure 4. In the right panel of Figure 4, the change in the regression coefficients $\boldsymbol{\beta}$ as a function of order M is shown. Both the log-cumulative hazard function and the regression coefficients obtained from `mlt()` are stable for $M \geq 10$ and very similar to the results one obtains from `coxph()`. This result shows that a simple yet fully parametric model produces practically equivalent results when compared to the semiparametric partial likelihood approach. There is no harm in choosing M "too large", except longer computing times of course. In this sense, there is no need to "tune" M.

A comparison with the Royston and Parmar (2002) spline model as implemented in the **flexsurv** package (Jackson 2017) shows that the two spline parameterisations of the log-cumulative hazard function h_Y are also practically equivalent (see Figure 5)

```
R> kn <- log(support(GBSG2y)$y)
R> fss_GBSG2_1 <- flexsurvspline(fm_GBSG2, data = GBSG2, scale = "hazard",
+                                k = 9, bknots = kn)
R> logLik(fss_GBSG2_1)
```

```
'log Lik.' -2555.856 (df=20)
```

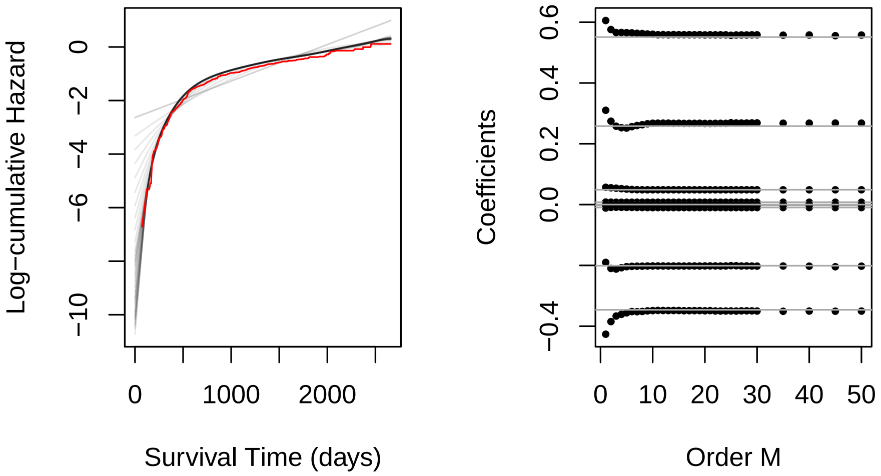

Figure 4: GBSG-2 Trial. Comparison of exact and partial likelihood for order $M = 1, \dots, 30, 35, 40, 45, 50$ of the Bernstein polynom approximating the log-cumulative hazard function h_Y. In the left panel the estimated log-cumulative hazard functions for varying M obtained by `mlt()` are shown in grey and the Nelson-Aalen-Breslow estimator obtained from `coxph()` in red. The right panel shows the trajectories of the regression coefficients $\boldsymbol{\beta}$ obtained for varying M from `mlt()` as dots. The horizonal lines represent the partial likelihood estimates from `coxph()`. This figure reproduces Figure 6 in Hothorn *et al.* (2017c).

```
R> logLik(mlt_GBSG2_1)

'log Lik.' -2559.151 (df=20)

R> cf <- coef(coxph_GBSG2_1)
R> cbind(coxph = cf, mlt = coef(mlt_GBSG2_1)[names(cf)],
+        fss = coef(fss_GBSG2_1)[names(cf)])

                     coxph          mlt           fss
horThyes     -0.3462416209 -0.348707936 -0.3473363301
age          -0.0094534071 -0.009932002 -0.0097895840
menostatPost  0.2581565452  0.267642672  0.2672204934
tsize         0.0077983290  0.007765951  0.0079051626
tgrade.L      0.5510838051  0.560223057  0.5554284914
tgrade.Q     -0.2011060204 -0.201942423 -0.2078499591
pnodes        0.0487818043  0.048761983  0.0485933664
progrec      -0.0022174942 -0.002210539 -0.0022112350
estrec        0.0001978178  0.000182990  0.0001891311
```

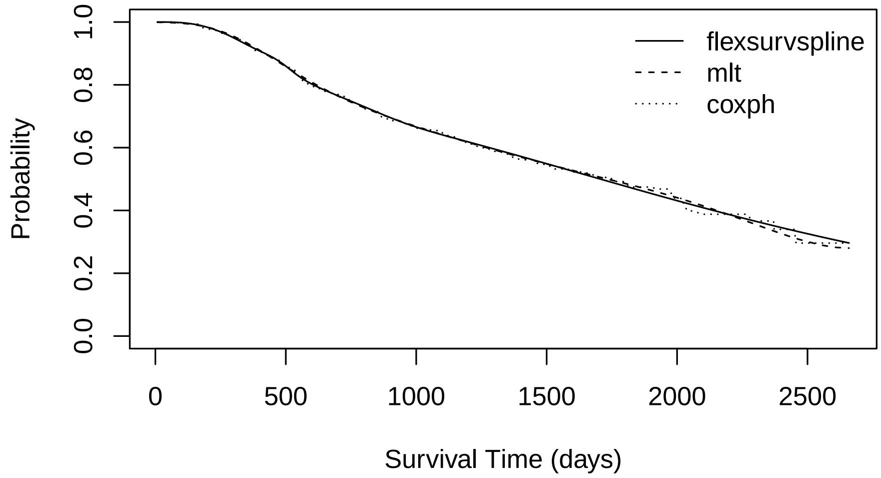

Figure 5: GBSG-2 Trial. Estimated survivor functions for patient 1 by the most likely transformation model (MLT) and the Royston and Parmar (2002) spline model fitted using `flexsurvspline()` as well as based on the Nelson-Aalen-Breslow estimator from `coxph()`.

Accelerated Failure Time (AFT) models arise when one restricts the baseline transformation $h_Y(y)$ to a possibly scaled log-transformation. With $h_Y(y) = \vartheta_1 \log(y) + \vartheta_2$ (with $\vartheta_1 \equiv 1$) and $F_Z = F_{\text{MEV}}$ the exponential AFT model arises which can be specified as the Cox model above, only the Bernstein basis for time needs to be replaced by a log-transformation and the corresponding parameter is restricted to one in the estimation

```
R> ly <- log_basis(GBSG2y, ui = "increasing")
R> ctm_GBSG2_2 <- ctm(ly, shifting = fm_GBSG2[-2L], data = GBSG2,
+                     negative = TRUE, todistr = "MinExtrVal")
R> mlt_GBSG2_2 <- mlt(ctm_GBSG2_2, data = GBSG2, fixed = c("log(y)" = 1),
+                     scale = TRUE)
```

The intercept ϑ_2 is contained in the log basis function `ly` and not in the shift term. The `survreg()` (package **survival**) and `phreg()` (package **eha**, Broström 2017) functions fit the same model, the results are again equivalent up to the sign of the regression coefficients

```
R> survreg_GBSG2_2 <- survreg(fm_GBSG2, data = GBSG2, dist = "exponential")
R> phreg_GBSG2_2 <- phreg(fm_GBSG2, data = GBSG2, dist = "weibull",
+                         shape = 1)
R> logLik(survreg_GBSG2_2)

'log Lik.' -2599.383 (df=10)

R> logLik(phreg_GBSG2_2)
```

```
'log Lik.' -2599.383 (df=10)
```

```
R> logLik(mlt_GBSG2_2)
```

```
'log Lik.' -2599.383 (df=10)
```

```
R> cbind(survreg = coef(survreg_GBSG2_2)[names(cf)],
+        phreg = -coef(phreg_GBSG2_2)[names(cf)],
+        mlt = coef(mlt_GBSG2_2)[names(cf)])
```

```
                  survreg         phreg           mlt
horThyes      0.3321617406  0.3321617406  0.3321614005
age           0.0094196671  0.0094196671  0.0094196280
menostatPost -0.2685358867 -0.2685358867 -0.2685364763
tsize        -0.0073179443 -0.0073179443 -0.0073180053
tgrade.L     -0.5193521191 -0.5193521191 -0.5193491789
tgrade.Q      0.2139217558  0.2139217558  0.2139210801
pnodes       -0.0461664688 -0.0461664688 -0.0461663602
progrec       0.0020670894  0.0020670894  0.0020671109
estrec       -0.0001788501 -0.0001788501 -0.0001788297
```

If we allow a scaled log-transformation $h_Y(y) = \vartheta_1 \log(y) + \vartheta_2$ (the intercept ϑ_2 is included in the baseline transformation), the resulting Weibull AFT model is fitted by `mlt()`, `survreg()` and `phreg()` using

```
R> mlt_GBSG2_3 <- mlt(ctm_GBSG2_2, data = GBSG2, scale = TRUE)
R> survreg_GBSG2_3 <- survreg(fm_GBSG2, data = GBSG2, dist = "weibull")
R> phreg_GBSG2_3 <- phreg(fm_GBSG2, data = GBSG2, dist = "weibull")
R> logLik(survreg_GBSG2_3)
```

```
'log Lik.' -2579.695 (df=11)
```

```
R> logLik(phreg_GBSG2_3)
```

```
'log Lik.' -2579.695 (df=11)
```

```
R> logLik(mlt_GBSG2_3)
```

```
'log Lik.' -2579.695 (df=11)
```

```
R> cbind(survreg = coef(survreg_GBSG2_3)[names(cf)] / survreg_GBSG2_3$scale,
+        phreg = - coef(phreg_GBSG2_3)[names(cf)],
+        mlt = coef(mlt_GBSG2_3)[names(cf)])
```

```
                  survreg         phreg           mlt
horThyes      0.3730909205  0.3730909205  0.3730890683
age           0.0094799262  0.0094799262  0.0094802554
menostatPost -0.2709027409 -0.2709027409 -0.2709161321
tsize        -0.0080149047 -0.0080149047 -0.0080152960
tgrade.L     -0.5727992624 -0.5727992624 -0.5727666451
tgrade.Q      0.2050597169  0.2050597169  0.2050384760
pnodes       -0.0528016503 -0.0528016503 -0.0528004271
progrec       0.0022848557  0.0022848557  0.0022847813
estrec       -0.0002484315 -0.0002484315 -0.0002483986
```

The estimated scale parameters $\hat{\vartheta}_1$ are 1.3903 (`survreg()`), 1.3903 (`phreg()`) and 1.3903 (`mlt()`).

Non-normal Linear Regression: Boston Housing Data The Boston Housing data are a prominent test-bed for parametric and non-parametric alternatives to a normal linear regression model. Assuming a conditional normal distribution for the median value of owner-occupied homes (medv, in USD 1000's, we use the corrected version) in the normal linear model with constant variance

$$\text{medv} \mid \boldsymbol{X} = \boldsymbol{x} \sim \text{N}(\tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}, \sigma^2)$$

as implemented in `lm()` is rather restrictive

```
R> data("BostonHousing2", package = "mlbench")
R> lm_BH <- lm(cmedv ~ crim + zn + indus + chas + nox + rm + age + dis +
+               rad + tax + ptratio + b + lstat, data = BostonHousing2)
```

We relax the model formulation without sacrificing the simplicity of a linear predictor of the explanatory variables in the linear transformation model

$$\mathbb{P}(\text{medv} \le y \mid \boldsymbol{X} = \boldsymbol{x}) = \Phi(h_Y(y) - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}) = \Phi(\boldsymbol{a}_{\text{Bs},6}(y)^\top \boldsymbol{\vartheta}_1 - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta})$$

and estimate *all* model parameters $(\boldsymbol{\vartheta}_1, \boldsymbol{\beta})$ simultaneously, *i.e.* both the regression coefficients *and* the baseline transformation h_Y. Because it is straightforward to evaluate the conditional distribution function, the likelihood can deal with right-censored `medvc` observations (≥ 50). This censoring was mostly ignored in other parametric or non-parametric analyses of this data set.

We start with a suitable definition of median value of owner-occupied homes, set-up a `Surv` object for this response and a model formula

```
R> BostonHousing2$medvc <- with(BostonHousing2, Surv(cmedv, cmedv < 50))
R> var_m <- numeric_var("medvc", support = c(10.0, 40.0), bounds = c(0, Inf))
R> fm_BH <- medvc ~ crim + zn + indus + chas + nox + rm + age +
+                     dis + rad + tax + ptratio + b + lstat
```

First, we aim at fitting a normal linear model taking censored observations properly into account. With linear baseline transformation h_Y, *i.e.* a Bernstein polynomial of order one or a linear function with intercept and positive slope, the model is equivalent to the model underlying `lm()` as explained in the introduction. Only the likelihood changes when we fit the model via

```
R> B_m <- polynomial_basis(var_m, coef = c(TRUE, TRUE),
+                          ui = matrix(c(0, 1), nrow = 1), ci = 0)
R> ctm_BH <- ctm(B_m, shift = fm_BH[-2L], data = BostonHousing2,
+                todistr = "Normal")
R> lm_BH_2 <- mlt(ctm_BH, data = BostonHousing2, scale = TRUE)
R> logLik(lm_BH_2)

'log Lik.' -1496.301 (df=15)
```

In a second step, we are interested in a possibly non-linear transformation of the response and use a Bernstein polynomial of order six. In principle, this approach is equivalent to using a Box-Cox transformation but with a more flexible transformation function. In a sense, we don't need to worry too much about the error distribution F_Z as only the additivity assumption on our linear predictor depends on this choice (which may or may not be a strong assumption!). The conditional transformation model is now given by this transformation of the response, a linear predictor of the explanatory variables the model and the normal distribution function; the `mlt()` function fits the model to the data

```
R> B_m <- Bernstein_basis(var_m, order = 6, ui = "increasing")
R> ctm_BH <- ctm(B_m, shift = fm_BH[-2L], data = BostonHousing2,
+                todistr = "Normal")
R> mlt_BH <- mlt(ctm_BH, data = BostonHousing2, scale = TRUE)
R> logLik(mlt_BH)

'log Lik.' -1324.698 (df=20)
```

The model can be compared with a normal linear model (fitted by `lm()`) on the scale of the fitted conditional distribution functions. Figure 6 shows the fitted values, *i.e.* the linear predictor $\tilde{\boldsymbol{x}}_i^\top \hat{\boldsymbol{\beta}}_N$ for each observation, and the observed response overlayed with the conditional distribution function for the corresponding observations. For the normal linear model featuring a *linear* baseline transformation h_Y, the fit seems appropriate for observations with linear predictor less than 30. For larger values, the linear predictor underestimates the observations. The conditional distribution obtained from the linear transformation model captures observations with large values of the response better. For smaller values of the response, the fit resembles the normal linear model, although with a smaller conditional variance.

Truncated Regression: Panel Study of Income Dynamics Mroz (1987) analysed the University of Michigan Panel Study of Income Dynamics (PSID) for the year 1975 (interview year 1976). The data consists of 753 married white women between the ages of 30 and 60 in 1975, with 428 working at some time during the year. The dependent variable is the wife's annual hours of work and we are interested in modelling the distribution of hours of work conditional on participation in the labour force, *i.e.* more than zero hours of work in 1975. We first set-up a subset consisting of those who actually worked (for money!) in 1975 and a model formula

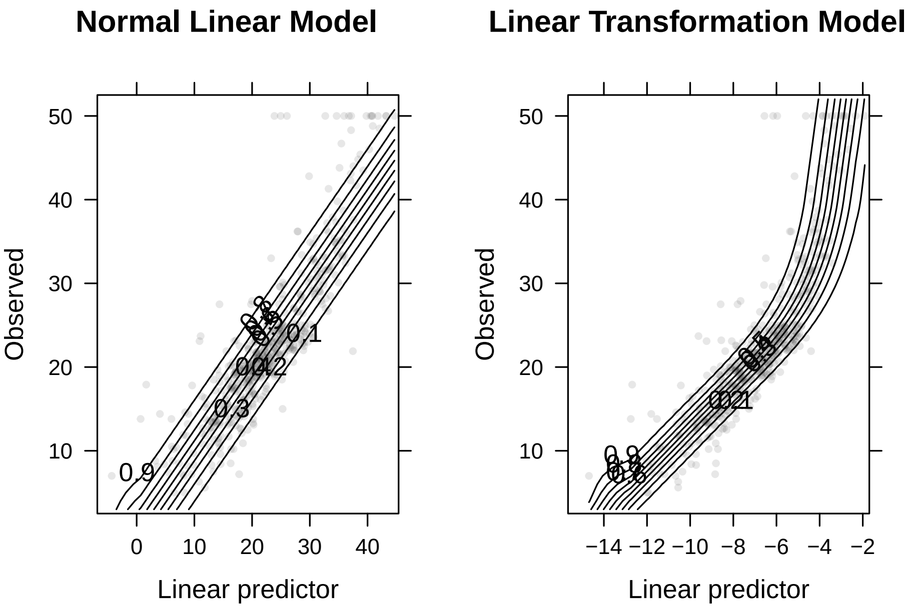

Figure 6: Boston Housing. Predicted vs. observed for the normal linear model (left) and the linear transformation model with smooth baseline transformation (right). The observations are overlayed with the conditional quantiles of the response given the linear predictor.

```
R> data("PSID1976", package = "AER")
R> PSID1976$nwincome <- with(PSID1976, (fincome - hours * wage)/1000)
R> PSID1976$hours <- as.double(PSID1976$hours)
R> PSID1976_0 <- subset(PSID1976, participation == "yes")
R> fm_PSID1976 <- hours ~ nwincome + education + experience +
+                         I(experience^2) + age + youngkids + oldkids
```

We use a linear regression model for left-truncated data and compare the results to `truncreg()` from package **truncreg** (Croissant and Zeileis 2016)

```
R> tr_PSID1976 <- truncreg(fm_PSID1976, data = PSID1976_0)
```

We use the `R()` function (see Appendix C.1) to specify left-truncation at zero, set-up a linear transformation function with positive slope for the response (we want to stay within the normal family) and a shift term

```
R> PSID1976_0$hours <- R(PSID1976_0$hours, tleft = 0)
R> b_hours <- as.basis(~ hours, data = PSID1976,
+                      ui = matrix(c(0, 1), nr  = 1), ci = 0)
R> ctm_PSID1976_1 <- ctm(b_hours, shift = fm_PSID1976[-2L],
+                        data = PSID1976_0, todistr = "Normal")
R> mlt_PSID1976_1 <- mlt(ctm_PSID1976_1, data = PSID1976_0, scale = TRUE)
```

The `mlt()` function does a slightly better job at maximising the likelihood than `truncreg()` which explains the differences in the estimated coefficients

```
R> logLik(tr_PSID1976)

'log Lik.' -3391.478 (df=9)

R> logLik(mlt_PSID1976_1)

'log Lik.' -3390.648 (df=9)

R> cf <- coef(mlt_PSID1976_1)
R> cbind(truncreg = coef(tr_PSID1976),
+        mlt = c(-cf[-grep("hours", names(cf))], 1) / cf["hours"])

                    truncreg          mlt
(Intercept)     2055.7127740 2123.5661425
nwincome          -0.5011526    0.1536498
education        -31.2696497  -29.8512046
experience        73.0066075   72.6100938
I(experience^2)   -0.9695106   -0.9436354
age              -25.3359778  -27.4435844
youngkids       -318.8521247 -484.7014006
oldkids          -91.6195316 -102.6517073
sigma            822.4792934  850.7631886
```

Of course, we might want to question the normal assumption by allowing a potentially non-linear transformation function. We simply change the linear to a non-linear baseline transformation h_Y at the price of five additional parameters in the model

```
R> var_h <- numeric_var("hours", support = range(PSID1976_0$hours$exact),
+                       bounds = c(0, Inf))
R> B_hours <- Bernstein_basis(var_h, order = 6, ui = "increasing")
R> ctm_PSID1976_2 <- ctm(B_hours, shift = fm_PSID1976[-2L],
+                        data = PSID1976_0, todistr = "Normal")
R> mlt_PSID1976_2 <- mlt(ctm_PSID1976_2, data = PSID1976_0,
+                        scale = TRUE)
R> logLik(mlt_PSID1976_2)

'log Lik.' -3375.477 (df=14)
```

and there might be a small advantage with the non-normal model.

2.3. Stratified Linear Transformation Models

Stratification in linear transformation models refers to a strata-specific transformation function but a shift term those regression coefficients are constant across strata. The model then reads

$$\mathbb{P}(Y \leq y \mid \text{stratum} = s, \boldsymbol{X} = \boldsymbol{x}) = F_Z(h(y \mid s, \boldsymbol{x})) = F_Z(h_Y(y \mid s) - \tilde{\boldsymbol{x}}^\top \boldsymbol{\beta}) = F_Z(\boldsymbol{c}(y, s, \boldsymbol{x})^\top \boldsymbol{\vartheta})$$

with basis function $\boldsymbol{c} = (\boldsymbol{a}^\top \otimes \boldsymbol{b}_{\text{stratum}}^\top, -\boldsymbol{b}_{\text{shift}}^\top)^\top$. The basis function $\boldsymbol{b}_{\text{stratum}}^\top$ is a dummy coding for the stratum variable and is defined using the `interacting` argument of `ctm()`. The constraints for the parameters of $\boldsymbol{a}$ have to be met for each single stratum, *i.e.* the total number of linear constraints is the number of constraints for $\boldsymbol{a}$ multiplied by the number of strata.

Discrete Responses

Categorical Data Analysis: Chinese Survey (Cont'd) We first estimate the distribution of happiness given health without taking any other explanatory variables into account, *i.e.* by treating health as a stratum variable (with dummy coding) but without any regression coefficients $\boldsymbol{\beta}$ in the model

```
R> b_health <- as.basis(~ R_health - 1, data = CHFLS)
R> ctm_CHFLS_3 <- ctm(b_happy, interacting = b_health, todist = "Logistic")
R> mlt_CHFLS_3 <- mlt(ctm_CHFLS_3, data = CHFLS, scale = TRUE)
R> logLik(mlt_CHFLS_3)
```

```
'log Lik.' -1192.226 (df=15)
```

```
R> predict(mlt_CHFLS_3, newdata = mkgrid(mlt_CHFLS_3), type = "distribution")
```

```
               R_health
R_happy              Poor   Not good        Fair        Good    Excellent
  Very unhappy   0.1999982 0.05035934 0.008676623 0.001718219 4.019721e-08
  Not too happy  0.6000123 0.38129503 0.154012410 0.073883591 7.602338e-02
  Somewhat happy 0.8999994 0.93525161 0.913232195 0.862542204 5.614046e-01
  Very happy     1.0000000 1.00000000 1.000000000 1.000000000 1.000000e+00
```

The conditional distribution for happiness given each health category is returned by `predict()` as a matrix. There is a clear tendency of people being happier with better health. We now 'adjust' for age and income by including a linear shift term which is constant across strata

```
R> ctm_CHFLS_4 <- ctm(b_happy, interacting = b_health, shifting = b_R,
+                     todist = "Logistic")
R> mlt_CHFLS_4 <- mlt(ctm_CHFLS_4, data = CHFLS, scale = TRUE)
R> coef(mlt_CHFLS_4)[c("R_age", "R_income")]
```

```
       R_age     R_income
0.0117382832 0.0002492679
```

Because the shift basis `b_R` is negative, the effects of both age and income on the happiness distribution are towards larger values of happiness, *i.e.* older and richer people are happier for all health levels (this is, of course, due to the restrictive model not allowing interactions between health and the other two variables). The "health-adjusted" log-odds ratios are now 1.0118 for each year of age and 1.0002 for each additional Yuan earned and, conditional on health, people are getting happier as they get older *and* richer.

Continuous Responses

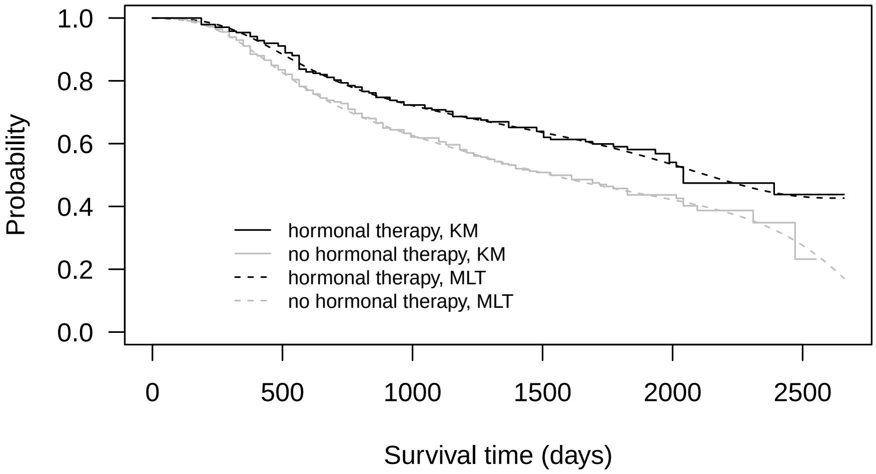

Figure 7: GBSG-2 Trial. Estimated survivor functions by the most likely transformation model (MLT) and the Kaplan-Meier (KM) estimator in the two treatment groups. The plot reproduces Figure 4 (left panel) in Hothorn *et al.* (2017c).

Survival Analysis: GBSG-2 Trial (Cont'd) The Cox model presented in Section 2.2 features one baseline function for all observations, an assumption which we are now going to relax. As a first simple example, we want to estimate two separate survivor functions for the two treatment regimes in the model

$$(F_{\text{MEV}}, (\boldsymbol{a}_{\text{Bs},10}(y)^\top \otimes (\mathbb{1}(\text{hormonal therapy}), 1 - \mathbb{1}(\text{hormonal therapy})))^\top, (\boldsymbol{\vartheta}_1^\top, \boldsymbol{\vartheta}_2^\top)^\top)$$

Here, the transformation functions $\boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_1$ and $\boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_2$ correspond to the untreated and treated groups, respectively

```
R> b_horTh <- as.basis(GBSG2$horTh)
R> ctm_GBSG2_4 <- ctm(B_GBSG2y, interacting = b_horTh,
+                     todistr = "MinExtrVal")
R> mlt_GBSG2_4 <- mlt(ctm_GBSG2_4, data = GBSG2)
```

The two survivor functions, along with the corresponding Kaplan-Meier estimates, are shown in Figure 7, the low-dimensional Bernstein polynomials produce a nicely smoothed version of the Kaplan-Meier step-functions.

In a second step, we allow treatment-specific baseline hazard functions while estimating a constant age effect in the model

$$(F_{\text{MEV}}, (\boldsymbol{a}_{\text{Bs},10}(y)^\top \otimes (\mathbb{1}(\text{hormonal therapy}), 1 - \mathbb{1}(\text{hormonal therapy})), \text{age})^\top, (\boldsymbol{\vartheta}_1^\top, \boldsymbol{\vartheta}_2^\top, \beta)).$$

This model is fitted by

```
R> ctm_GBSG2_5 <- ctm(B_GBSG2y, interacting = b_horTh, shifting = ~ age,
+                      data = GBSG2, todistr = "MinExtrVal")
R> mlt_GBSG2_5 <- mlt(ctm_GBSG2_5, data = GBSG2, scale = TRUE)
```

The corresponding stratified Cox model with parameter estimation based on the partial likelihood is

```
R> coxph_GBSG2_5 <- coxph(Surv(time, cens) ~ age + strata(horTh),
+                          data = GBSG2)
R> cf <- coef(coxph_GBSG2_5)
R> cbind(coxph = cf, mlt = coef(mlt_GBSG2_5)[names(cf)])
```

```
            coxph          mlt
age -0.0003695703 -0.000464507
```

The Cox model fitted via the stratified partial likelihood (`coxph()`) and the transformation model agree on a positive age effect $\hat{\beta}$, *i.e.* older patients seem to survive longer (note that `coxph()` estimates a positive shift effect as does the transformation model specified above).

2.4. Conditional Transformation Models

The most complex class of models currently supported by the `ctm()` function allows basis functions of the form

$$\boldsymbol{c} = (\boldsymbol{a}^\top \otimes (\boldsymbol{b}_1^\top, \dots, \boldsymbol{b}_J^\top), -\boldsymbol{b}_{\text{shift}}^\top).$$

The model may include response-varying coefficients (as defined by the basis $\boldsymbol{a}$) corresponding to the bases $(\boldsymbol{b}_1^\top, \dots, \boldsymbol{b}_J^\top)$ and constant shift effects ($\boldsymbol{b}_{\text{shift}}$). Such a model is set-up using `ctm()` with arguments `response` (basis $\boldsymbol{a}$), `interacting` (basis $(\boldsymbol{b}_1^\top, \dots, \boldsymbol{b}_J^\top)$) and `shifting` (basis $-\boldsymbol{b}_{\text{shift}}^\top$). It would be conceptually possible to fit even more complex conditional transformation models of the form

$$\boldsymbol{c} = (\boldsymbol{a}_1^\top \otimes \boldsymbol{b}_1^\top, \dots, \boldsymbol{a}_J^\top \otimes \boldsymbol{b}_J^\top)$$

with a less restrictive user interface.

Discrete Responses

Categorical Data Analysis: Chinese Survey (Cont'd) In a series of non-proportional odds models for the conditional distribution of happiness we study the impact of health, age and income on happiness. Similar to the stratified model `ctm_CHFLS_3`, we estimate the conditional distribution of happiness separately for each health level, but instead of using a dummy coding we use treatment contrasts

```
R> contrasts(CHFLS$R_health) <- "contr.treatment"
R> b_health <- as.basis(~ R_health, data = CHFLS)
R> ctm_CHFLS_5 <- ctm(b_happy, interacting = b_health, todist = "Logistic")
R> mlt_CHFLS_5 <- mlt(ctm_CHFLS_5, data = CHFLS, scale = TRUE)
R> predict(mlt_CHFLS_5, newdata = mkgrid(mlt_CHFLS_5), type = "distribution")
```

```
                R_health
R_happy                Poor   Not good        Fair        Good    Excellent
  Very unhappy   0.1998996 0.05039632 0.008665935 0.001721158 1.013586e-06
  Not too happy  0.5998795 0.38129624 0.153996989 0.073882498 7.602025e-02
  Somewhat happy 0.8997576 0.93522510 0.913243808 0.862559778 5.614392e-01
  Very happy     1.0000000 1.00000000 1.000000000 1.000000000 1.000000e+00
```

```
R> logLik(mlt_CHFLS_5)
```

```
'log Lik.' -1192.226 (df=15)
```

The log-likelihood and the fitted distribution are, of course, equivalent but the parameters allow a direct interpretation of the effect of health relative to the baseline category `Poor`. In a second step, we fit a non-proportional odds model where the effects of age and income are allowed to vary with happiness

```
R> b_R <- as.basis(~ R_age + R_income, data = CHFLS, remove_intercept = TRUE,
+                    scale = TRUE)
R> ctm_CHFLS_6 <- ctm(b_happy, interacting = b_R, todist = "Logistic")
R> mlt_CHFLS_6 <- mlt(ctm_CHFLS_6, data = CHFLS, scale = TRUE)
R> logLik(mlt_CHFLS_6)
```

```
'log Lik.' -1472.42 (df=6)
```

and finally we include all three variables (health, age and income) allowing happiness-varying effects as

```
R> ctm_CHFLS_7 <- ctm(b_happy, interacting = c(h = b_health, R = b_R),
+       todist = "Logistic")
R> mlt_CHFLS_7 <- mlt(ctm_CHFLS_7, data = CHFLS, scale = TRUE)
R> logLik(mlt_CHFLS_7)
```

```
'log Lik.' -1182.779 (df=21)
```

Categorical Data Analysis: Iris Data For an unordered response in a multi-class problem, the conditional distribution can be estimated using a multinomial regression. In a non-proportional odds model allowing response-specific regression coefficients, the ordering of the response levels only affects the corresponding regression coefficients but the fitted density is invariant with respect to the ordering applied as a comparison with `multinom()` from package **nnet** (Venables and Ripley 2002; Ripley 2016) for the iris data shows

```
R> fm_iris <- Species ~ Sepal.Length + Sepal.Width +
+                       Petal.Length + Petal.Width
R> multinom_iris <- nnet::multinom(fm_iris, data = iris, trace = FALSE)
R> logLik(multinom_iris)
```

```
'log Lik.' -5.949867 (df=10)
```

```
R> iris$oSpecies <- ordered(iris$Species)
R> b_Species <- as.basis(iris$oSpecies)
R> b_x <- as.basis(fm_iris[-2L], data = iris, scale = TRUE)
R> ctm_iris <- ctm(b_Species, interacting = b_x,
+                  todistr = "Logistic")
R> mlt_iris <- mlt(ctm_iris, data = iris, scale = TRUE)
R> logLik(mlt_iris)

'log Lik.' -5.95478 (df=10)

R> p1 <- predict(mlt_iris, newdata = iris, q = sort(unique(iris$oSpecies)),
+               type = "density")
R> p2 <- predict(multinom_iris, newdata = iris, type = "prob")
R> max(abs(t(p1) - p2))

[1] 0.01222766
```

From this point of view, the multinomial model for an unordered categorical response is equivalent to a non-proportional odds model with response-varying effects.

Continuous Responses

Survival Analysis: GBSG-2 Trial (Cont'd) The Cox model for the comparison of the survivor distribution between the untreated and treated group assuming proportional hazards, *i.e.* the model $(F_{\text{MEV}}, (\boldsymbol{a}_{\text{Bs},10}^\top, \mathbb{1}(\text{hormonal therapy}))^\top, (\boldsymbol{\vartheta}_1^\top, \beta)^\top)$, implements the transformation function $h(y \mid \text{treatment}) = \boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_1 + \mathbb{1}(\text{hormonal therapy})\beta$ where $\boldsymbol{a}_{\text{Bs},10}^\top \boldsymbol{\vartheta}_1$ is the log-cumulative baseline hazard function parameterised by a Bernstein polynomial and $\beta \in \mathbb{R}$ is the log-hazard ratio of hormonal therapy

```
R> ctm_GBSG2_6 <- ctm(B_GBSG2y, shifting = ~ horTh, data = GBSG2,
+                     todistr = "MinExtrVal")
R> mlt_GBSG2_6 <- mlt(ctm_GBSG2_6, data = GBSG2)
R> logLik(mlt_GBSG2_6)

'log Lik.' -2607.361 (df=12)
```

This is the classical Cox model with one treatment parameter β but fully parameterised baseline transformation function which was fitted by the exact log-likelihood under ten linear constraints. The model assumes proportional hazards, an assumption whose appropriateness we want to assess using the non-proportional hazards model $(F_{\text{MEV}}, (\boldsymbol{a}_{\text{Bs},10}^\top \otimes (1, \mathbb{1}(\text{hormonal therapy})))^\top, \boldsymbol{\vartheta})$ with transformation function

$$h(y \mid \text{treatment}) = \boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_1 + \mathbb{1}(\text{hormonal therapy})\boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_2.$$

The function $\boldsymbol{a}_{\text{Bs},10}^\top \boldsymbol{\vartheta}_2$ is the time-varying treatment effect and can be interpreted as the deviation, on the scale of the transformation function, induced by the hormonal therapy. Under the null hypothesis of no treatment effect, we would expect $\boldsymbol{\vartheta}_2 \equiv \mathbf{0}$. It should be noted that the log-cumulative hazard function $\boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_1$ must by monotone. The deviation function $\boldsymbol{a}_{\text{Bs},10}(y)^\top \boldsymbol{\vartheta}_2$ does not need to be monotone, however, the sum of the two functions needs to be monotone. Sums of such Bernstein polynomials with coefficients $\boldsymbol{\vartheta}_1$ and $\boldsymbol{\vartheta}_2$ are again Bernstein polynomials with coefficients $\boldsymbol{\vartheta}_1 + \boldsymbol{\vartheta}_2$. Thus, monotonicity of the sum can be implemented by monotonicity of $\boldsymbol{\vartheta}_1 + \boldsymbol{\vartheta}_2$. The argument `sumconstr = TRUE` implements the latter constraint (the default, in this specific situation). Figure 8 shows the time-varying treatment effect $\boldsymbol{a}_{\text{Bs},10}^\top \hat{\boldsymbol{\vartheta}}_{2,N}$, together with a 95% confidence band (see Section 5 for a description of the method). The 95% confidence interval around the log-hazard ratio $\hat{\beta}$ is plotted in addition and since the latter is fully covered by the confidence band for the time-varying treatment effect there is no reason to question the treatment effect computed under the proportional hazards assumption.

```
R> b_horTh <- as.basis(~ horTh, data = GBSG2)
R> ctm_GBSG2_7 <- ctm(B_GBSG2y, interacting = b_horTh,
+                     todistr = "MinExtrVal")
R> nd <- data.frame(y = GBSG2$time[1:2], horTh = unique(GBSG2$horTh))
R> attr(model.matrix(ctm_GBSG2_7, data = nd), "constraint")

$ui
20 x 22 sparse Matrix of class "dgCMatrix"

 [1,] -1  1  .  .  .  .  .  .  .  . .  .  .  .  .  .  .  .  .  .  . .
 [2,]  . -1  1  .  .  .  .  .  .  . .  .  .  .  .  .  .  .  .  .  . .
 [3,]  .  . -1  1  .  .  .  .  .  . .  .  .  .  .  .  .  .  .  .  . .
 [4,]  .  .  . -1  1  .  .  .  .  . .  .  .  .  .  .  .  .  .  .  . .
 [5,]  .  .  .  . -1  1  .  .  .  . .  .  .  .  .  .  .  .  .  .  . .
 [6,]  .  .  .  .  . -1  1  .  .  . .  .  .  .  .  .  .  .  .  .  . .
 [7,]  .  .  .  .  .  . -1  1  .  . .  .  .  .  .  .  .  .  .  .  . .
 [8,]  .  .  .  .  .  .  . -1  1  . .  .  .  .  .  .  .  .  .  .  . .
 [9,]  .  .  .  .  .  .  .  . -1  1 .  .  .  .  .  .  .  .  .  .  . .
[10,]  .  .  .  .  .  .  .  .  . -1 1  .  .  .  .  .  .  .  .  .  . .
[11,] -1  1  .  .  .  .  .  .  .  . . -1  1  .  .  .  .  .  .  .  . .
[12,]  . -1  1  .  .  .  .  .  .  . .  . -1  1  .  .  .  .  .  .  . .
[13,]  .  . -1  1  .  .  .  .  .  . .  .  . -1  1  .  .  .  .  .  . .
[14,]  .  .  . -1  1  .  .  .  .  . .  .  .  . -1  1  .  .  .  .  . .
[15,]  .  .  .  . -1  1  .  .  .  . .  .  .  .  . -1  1  .  .  .  . .
[16,]  .  .  .  .  . -1  1  .  .  . .  .  .  .  .  . -1  1  .  .  . .
[17,]  .  .  .  .  .  . -1  1  .  . .  .  .  .  .  .  . -1  1  .  . .
[18,]  .  .  .  .  .  .  . -1  1  . .  .  .  .  .  .  .  . -1  1  . .
[19,]  .  .  .  .  .  .  .  . -1  1 .  .  .  .  .  .  .  .  . -1  1 .
[20,]  .  .  .  .  .  .  .  .  . -1 1  .  .  .  .  .  .  .  .  . -1 1

$ci
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

R> mlt_GBSG2_7 <- mlt(ctm_GBSG2_7, data = GBSG2)
```

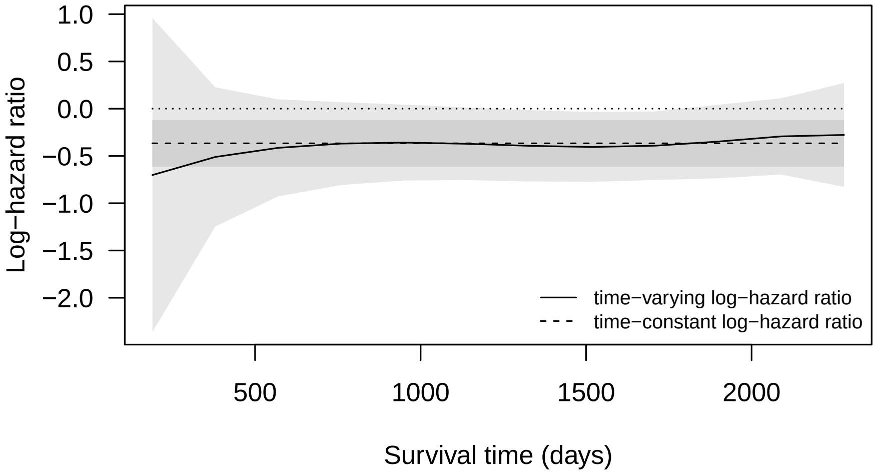

Figure 8: GBSG-2 Trial. Verification of proportional hazards: The log-hazard ratio $\hat{\beta}$ (dashed line) with 95% confidence interval (dark grey) is fully covered by a 95% confidence band for the time-varying treatment effect (light grey, the estimate is the solid line) computed from a non-proportional hazards model. The plot reproduces Figure 4 (right panel) in Hothorn *et al.* (2017c).

```
R> logLik(mlt_GBSG2_7)
```

```
'log Lik.' -2605.948 (df=22)
```

In a second step, we allow an age-varying treatment effect in the model $(F_{\text{MEV}}, (\boldsymbol{a}_{\text{Bs},10}(y)^\top \otimes (\mathbb{1}(\text{hormonal therapy}), 1 - \mathbb{1}(\text{hormonal therapy})) \otimes \boldsymbol{b}_{\text{Bs},3}(\text{age})^\top)^\top, \boldsymbol{\vartheta})$. For both treatment groups, we estimate a conditional transformation function of survival time y given age parameterised as the tensor basis of two Bernstein bases. Each of the two basis functions comes with 10×3 linear constraints, so the model was fitted under 60 linear constraints

```
R> var_a <- numeric_var("age", support = range(GBSG2$age))
R> B_age <- Bernstein_basis(var_a, order = 3)
R> b_horTh <- as.basis(GBSG2$horTh)
R> ctm_GBSG2_8 <- ctm(B_GBSG2y,
+                      interacting = b(horTh = b_horTh, age = B_age),
+                      todistr = "MinExtrVal")
R> mlt_GBSG2_8  <- mlt(ctm_GBSG2_8, data = GBSG2)
R> logLik(mlt_GBSG2_8)
```

```
'log Lik.' -2588.784 (df=88)
```

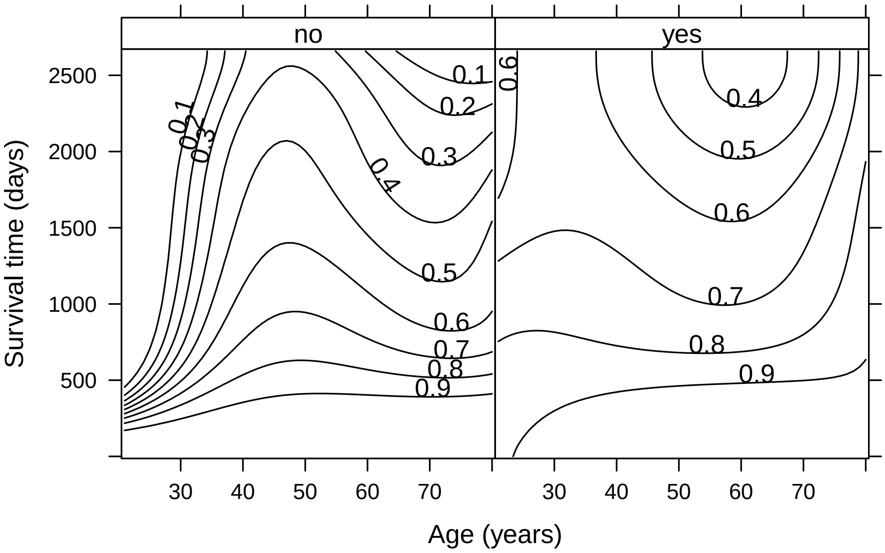

Figure 9: GBSG-2 Trial. Prognostic and predictive effect of age. The contours depict the conditional survivor functions given treatment and age of the patient. The plot reproduces Figure 5 in Hothorn *et al.* (2017c).

Figure 9 allows an assessment of the prognostic and predictive properties of age. As the survivor functions are clearly larger under hormonal treatment for all patients, the positive treatment effect applies to all patients. However, the size of the treatment effect varies greatly. For women younger than 30, the effect is most pronounced and levels-off a little for older patients. In general, the survival times are longest for women between 40 and 60 years old. Younger women suffer the highest risk; for women older than 60 years, the risk starts to increase again. This effect is shifted towards younger women by the application of hormonal treatment.

Quantile Regression: Head Circumference The Fourth Dutch Growth Study (Fredriks *et al.* 2000) is a cross-sectional study on growth and development of the Dutch population younger than 22 years. Stasinopoulos and Rigby (2007) fitted a growth curve to head circumferences (HC) of 7040 boys using a GAMLSS model with a Box-Cox t distribution describing the first four moments of head circumference conditionally on age. The model showed evidence of kurtosis, especially for older boys. We fit the same growth curves by the conditional transformation model $(\Phi, (\boldsymbol{a}_{\text{Bs},3}(\text{HC})^\top \otimes \boldsymbol{b}_{\text{Bs},3}(\text{age}^{1/3})^\top)^\top, \boldsymbol{\vartheta})$ by maximisation of the approximate log-likelihood under 3×4 linear constraints

```
R> data("db", package = "gamlss.data")
R> db$lage <- with(db, age^(1/3))
R> var_head <- numeric_var("head", support = quantile(db$head, c(.1, .9)),
+                          bounds = range(db$head))
R> B_head <- Bernstein_basis(var_head, order = 3, ui = "increasing")
```

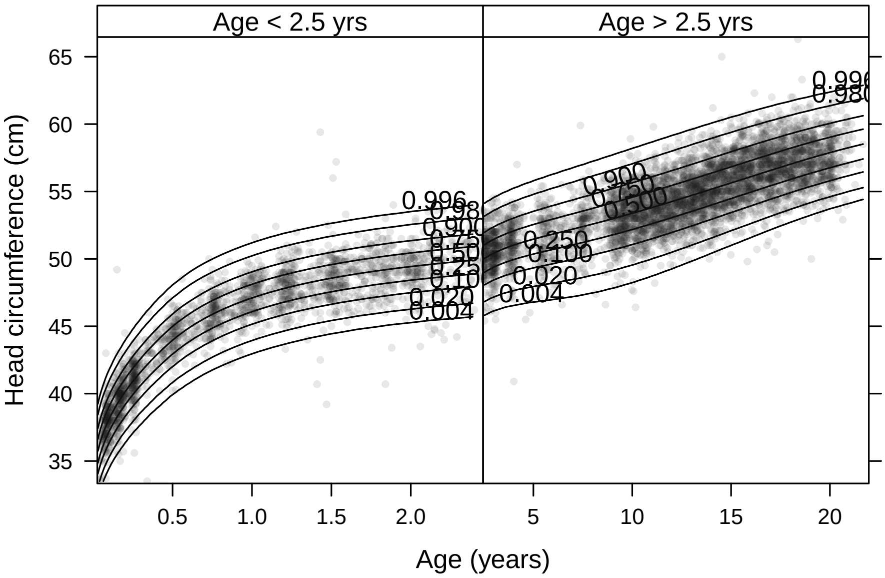

Figure 10: Head Circumference Growth. Observed head circumference and age for 7040 boys with estimated quantile curves for $\tau = 0.04, 0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98, 0.996$. The plot reproduces Figure 3 in Hothorn *et al.* (2017c).

```
R> var_lage <- numeric_var("lage", support = quantile(db$lage, c(.1, .9)),
+                              bounds = range(db$lage))
R> B_age <- Bernstein_basis(var_lage, order = 3, ui = "none")
R> ctm_head <- ctm(B_head, interacting = B_age)
R> mlt_head <- mlt(ctm_head, data = db, scale = TRUE)
```

Figure 10 shows the data overlaid with quantile curves obtained via inversion of the estimated conditional distributions. The figure very closely reproduces the growth curves presented in Figure 16 of Stasinopoulos and Rigby (2007) and also indicates a certain asymmetry towards older boys.

Non-normal Linear Regression: Boston Housing Data (Cont'd) A response-varying coefficient model, also called distribution regression (Foresi and Peracchi 1995; Chernozhukov *et al.* 2013; Koenker *et al.* 2013), for the Boston Housing data is

$$\begin{aligned}
\mathbb{P}(\text{medv} \le y \mid \boldsymbol{X} = \boldsymbol{x}) &= \Phi\left(h_Y(y) - \sum_{j=1}^{J} \boldsymbol{\beta}_j(y)\tilde{\boldsymbol{x}}_j\right) \\
&= \Phi\left(\boldsymbol{a}_{\text{Bs},6}(y)^\top \boldsymbol{\vartheta}_0 - \sum_{j=1}^{J} \boldsymbol{a}_{\text{Bs},6}(y)^\top \boldsymbol{\vartheta}_j \tilde{\boldsymbol{x}}_j\right) \\
&= \Phi\left(\boldsymbol{a}_{\text{Bs},6}(y)^\top \left(\boldsymbol{\vartheta}_0 - \sum_{j=1}^{J} \boldsymbol{\vartheta}_j \tilde{\boldsymbol{x}}_j\right)\right) \\
&= \Phi\left(\boldsymbol{a}_{\text{Bs},6}(y)^\top \left(\boldsymbol{\vartheta}_0 - \boldsymbol{\vartheta}(\boldsymbol{x})\right)\right)
\end{aligned}$$

The model requires the parameters $\boldsymbol{\vartheta}_0 - \boldsymbol{\vartheta}(\boldsymbol{x})$ to be monotone increasing for all possible values of $\boldsymbol{x}$. This type of constraint can be implemented using the `sumconstr = TRUE` argument to `ctm()`. The model is implemented using the basis function $\boldsymbol{c} = (\boldsymbol{a}_{\text{Bs},6}^\top \otimes (1, \tilde{\boldsymbol{x}}^\top))^\top$, the intercept is required here and $\tilde{\boldsymbol{x}}$ should be scaled to the unit interval. This model, here using parameters $\boldsymbol{\vartheta}_0 + \boldsymbol{\vartheta}(\boldsymbol{x})$, can be fitted using

```
R> b_BH_s <- as.basis(fm_BH[-2L], data = BostonHousing2, scale = TRUE)
R> ctm_BHi <- ctm(B_m, interacting = b_BH_s, sumconstr = TRUE)
R> mlt_BHi <- mlt(ctm_BHi, data = BostonHousing2)
R> logLik(mlt_BHi)

'log Lik.' -1224.453 (df=98)
```

This takes quite some time, simply because the number of constraints is is very large, depending on the number of explanatory variables. It might make sense to restrict all partial functions, and thus all partial parameters $\boldsymbol{\vartheta}_j$ to be monotone (argument `sumconstr = FALSE`):

```
R> ctm_BHi2 <- ctm(B_m, interacting = b_BH_s, sumconstr = FALSE)
R> mlt_BHi2 <- mlt(ctm_BHi2, data = BostonHousing2)
R> logLik(mlt_BHi2)

'log Lik.' -1274.368 (df=98)
```

Figure 11 compares the fitted densities for the linear transformation model with constant regression coefficients `mlt_BH` and the two distribution regression models `mlt_BHi` and `mlt_BHi2`. For some observations, the variance of the conditional distribution functions seems to be smaller in the more complex model distribution regression models with response-varying effects, compared to a linear transformation model with constant shift effects $\boldsymbol{\beta}$. There is not very much difference between the distribution regression models with different constraints.

Count Responses

Finally, we study a transformation model with response-varying coefficients for count data.

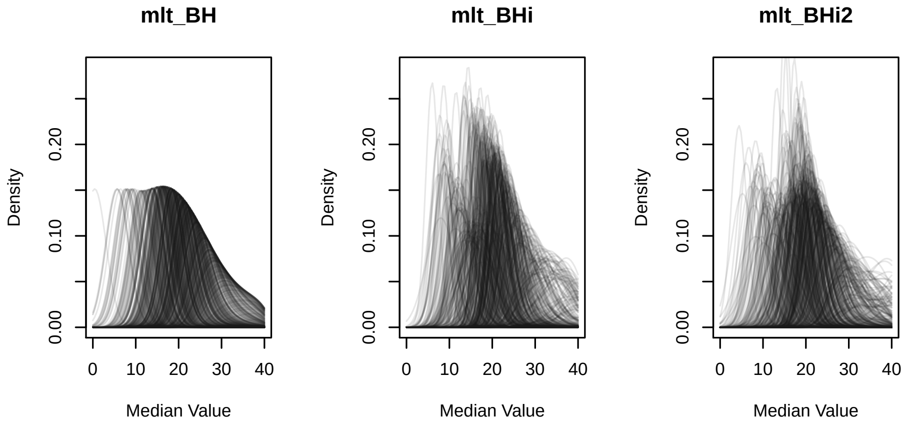

Figure 11: Boston Housing. Fitted conditional densities for the linear transformation model with constant regression coefficients (`mlt_BH`) and the response-varying coefficient models `mlt_BHi` and `mlt_BHi2`, the last model assumes monotone response-varying coefficient functions.

Analysis of Count Data: Tree Pipit Counts Müller and Hothorn (2004) reported data on the number of tree pipits *Anthus trivialis*, a small passerine bird, counted on 86 forest plots at a light gradient ranging from open and sunny stands (small cover storey) to dense and dark stands (large cover storey). We model the conditional distribution of the number of tree pipits at one plot given the cover storey at this plot by the transformation model $(\Phi, (\boldsymbol{a}^\top \otimes \boldsymbol{b}_{\text{Bs},4}(\text{cover storey})^\top)^\top, \boldsymbol{\vartheta})$, where $\boldsymbol{a}(y) = \boldsymbol{e}_5(y+1), y = 0, \dots, 4$; the model is fitted under 4×5 linear constraints. In this model for count data, the conditional distribution depends on both the number of counted birds and the cover storey and the effect of cover storey may change with different numbers of birds observed

```
R> data("treepipit", package = "coin")
R> treepipit$ocounts <- ordered(treepipit$counts)
R> B_cs <- Bernstein_basis(var = numeric_var("coverstorey", support = 1:110),
+                           order = 4)
R> B_c <- as.basis(treepipit$ocounts)
R> ctm_treepipit <- ctm(B_c, interacting = B_cs)
R> mlt_treepipit <- mlt(ctm_treepipit, data = treepipit, scale = TRUE,
+                       optim = mltoptim()["spg"])
```

The left panel of Figure 12 depicts the observations and the center panel shows the conditional distribution function evaluated for $0, \dots, 5$ observed birds. The conditional distribution function obtained from a generalised additive Poisson (GAM) model with smooth mean effect of cover storey (computed using **mgcv**, Wood 2006, 2017) is given in the right panel

```
R> gam_treepipit <- gam(counts ~ s(coverstorey), data = treepipit,
+                       family = "poisson")
```

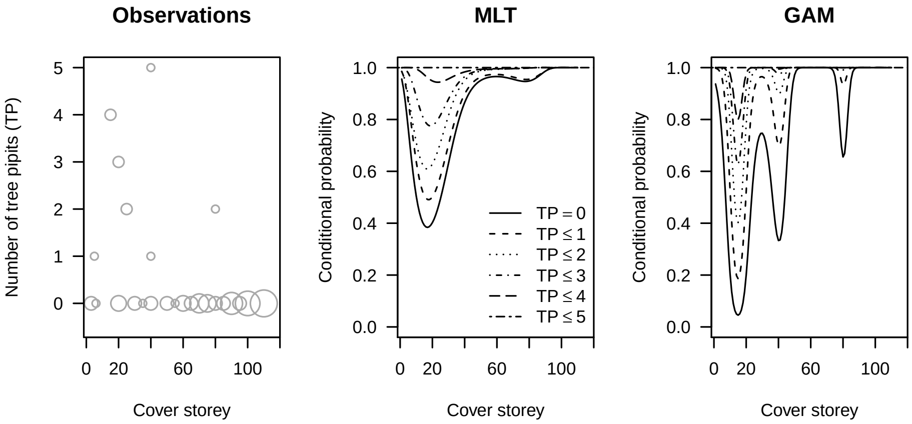

Figure 12: Tree Pipit Counts. Observations (left panel, the size of the points is proportional to the number of observations) and estimated conditional distribution of number of tree pipits given cover storey by the most likely transformation model (MLT, center panel) and a generalised additive Poisson model (function `gam()` in package **mgcv**, GAM, right panel). The plot reproduces Figure 7 in Hothorn *et al.* (2017c).

Despite some overfitting, this model is more restrictive than the transformation model because one mean function determines the whole distribution (the local minima of the conditional distributions as a function of cover storey are constant in the right panel whereas they are shifted towards higher values of cover storey in the center panel).

3. Most Likely Transformations

In this Section we review the underpinnings of the `mlt()` function implementing the *most likely transformation* estimator. Most of the material in this section originates from Hothorn *et al.* (2017b). For a given transformation function h, the likelihood contribution of a datum $C = (\underline{y}, \bar{y}] \in \mathfrak{C}$ is defined in terms of the distribution function (Lindsey 1996):

$$\mathcal{L}(h \mid Y \in C) := \int_C f_Y(y \mid h) d\mu(y) = F_Z(h(\bar{y})) - F_Z(h(\underline{y})).$$

This "exact" definition of the likelihood applies to most practically interesting situations and, in particular, allows discrete and (conceptually) continuous as well as censored or truncated observations C. For a discrete response y_k we have $\bar{y} = y_k$ and $\underline{y} = y_{k-1}$ such that $\mathcal{L}(h \mid Y = y_k) = f_Y(y_k \mid h) = F_Z(h(\bar{y})) - F_Z(h(\underline{y}))$. For absolutely continuous random variables Y we always practically observe an imprecise datum $(\underline{y}, \bar{y}] \subset \mathbb{R}$ and, for short intervals $(\underline{y}, \bar{y}]$, approximate the exact likelihood $\mathcal{L}(h \mid Y \in (\underline{y}, \bar{y}])$ by the term $(\bar{y} - \underline{y}) f_Y(y \mid h)$ or simply $f_Y(y \mid h)$ with $y = (\underline{y} + \bar{y})/2$ (Lindsey 1999). This approximation only works for relatively precise measurements, *i.e.* short intervals. If longer intervals are observed, one speaks of "censoring" and relies on the exact definition of the likelihood contribution instead of using the above approximation (Klein and Moeschberger 2003). In summary, the likelihood contribution of a conceptually "exact continuous" or left, right or interval-censored continuous or discrete observation $(\underline{y}, \bar{y}]$ is given by

$$\mathcal{L}(h \mid Y \in (\underline{y}, \bar{y}]) \begin{cases} \approx f_Z(h(y))h'(y) & y = (\underline{y} + \bar{y})/2 \in \Xi \quad \text{“‘‘exact continuous”’’} \\ = 1 - F_Z(h(\underline{y})) & y \in (\underline{y}, \infty) \cap \Xi \quad \text{‘right-censored’} \\ = F_Z(h(\bar{y})) & y \in (-\infty, \bar{y}] \cap \Xi \quad \text{‘left-censored’} \\ = F_Z(h(\bar{y})) - F_Z(h(\underline{y})) & y \in (\underline{y}, \bar{y}] \cap \Xi \quad \text{‘interval-censored’}, \end{cases}$$

under the assumption of random censoring. The likelihood is more complex under dependent censoring (Klein and Moeschberger 2003) but this is is not covered by the **mlt** implementation. The likelihood contribution $\mathcal{L}(h \mid Y \in (y_k, y_{k-1}])$ of an ordered factor in category y_k is equivalent to the term $\mathcal{L}(h \mid Y \in (\underline{y}, \bar{y}])$ contributed by an interval-censored observation $(\underline{y}, \bar{y}]$ when category y_k was defined by the interval $(\underline{y}, \bar{y}]$. Thus, the expression $F_Z(h(\bar{y})) - F_Z(h(\underline{y}))$ for the likelihood contribution reflects the equivalence of interval-censoring and categorisation at corresponding cut-off points.

For truncated observations in the interval $(y_l, y_r] \subset \Xi$, the above likelihood contribution is defined in terms of the distribution function conditional on the truncation

$$F_Y(y \mid Y \in (y_l, y_r]) = F_Z(h(y) \mid Y \in (y_l, y_r]) = \frac{F_Z(h(y))}{F_Z(h(y_r)) - F_Z(h(y_l))} \quad \forall y \in (y_l, y_r]$$

and thus the likelihood contribution changes to (Klein and Moeschberger 2003)

$$\frac{\mathcal{L}(h \mid Y \in (\underline{y}, \bar{y}])}{F_Z(h(y_r)) - F_Z(h(y_l))} = \frac{\mathcal{L}(h \mid Y \in (\underline{y}, \bar{y}])}{\mathcal{L}(h \mid Y \in (y_l, y_r])} \quad \text{when } y_l < \underline{y} < \bar{y} \le y_r.$$

It is important to note that the likelihood is always *defined* in terms of a distribution function (Lindsey 1999) and it therefore makes sense to directly model the distribution function of interest. The ability to uniquely characterise this distribution function by the transformation function h gives rise to the following definition of an estimator $\hat{h}_N$. For an independent sample of possibly censored or truncated observations $C_1, \dots, C_N$ from $\mathbb{P}_Y$ the estimator

$$\hat{h}_N := \operatorname*{arg\,max}_{\tilde{h} \in \mathcal{H}} \sum_{i=1}^{N} \log(\mathcal{L}(\tilde{h} \mid Y \in C_i))$$

is called the most likely transformation (MLT). In **mlt**, we parameterise the transformation function $h(y)$ as a linear function of its basis-transformed argument y using a basis function $\boldsymbol{a} : \Xi \to \mathbb{R}^P$ such that $h(y) = \boldsymbol{a}(y)^\top \boldsymbol{\vartheta}, \boldsymbol{\vartheta} \in \mathbb{R}^P$. The choice of the basis function $\boldsymbol{a}$ is problem-specific, practically relevant examples were discussed in Section 2. In the conditional case we use the basis $\boldsymbol{c}(y, \boldsymbol{x})$ instead of $\boldsymbol{a}(y)$. The likelihood $\mathcal{L}$ only requires evaluation of h, and only an approximation thereof using the Lebesgue density of “exact continuous” observations makes the evaluation of the first derivative of $h(y)$ with respect to y necessary. In this case, the derivative with respect to y is given by $h'(y) = \boldsymbol{a}'(y)^\top \boldsymbol{\vartheta}$ and we assume that $\boldsymbol{a}'$ is available. In the following we write $h = \boldsymbol{a}^\top \boldsymbol{\vartheta}$ and $h' = \boldsymbol{a}'^\top \boldsymbol{\vartheta}$ for the transformation function and its first derivative omitting the argument y and we assume that both functions are bounded away from $-\infty$ and ∞. For the basis functions discussed in Section 2, the constraint $\boldsymbol{\vartheta} \in \Theta$ can be written as $\boldsymbol{C}\boldsymbol{\vartheta} \ge \boldsymbol{0}$, thus the solution to the optimisation problem

$$\hat{\boldsymbol{\vartheta}}_N := \operatorname*{arg\,max}_{\boldsymbol{C}\boldsymbol{\vartheta} \ge \boldsymbol{0}} \sum_{i=1}^{N} \log(\mathcal{L}(\boldsymbol{a}^\top \boldsymbol{\vartheta} \mid Y \in C_i))$$

is the maximum likelihood estimator. The plug-in estimator for the most likely transformation is $\hat{h}_N := \boldsymbol{a}^\top \hat{\boldsymbol{\vartheta}}_N$.

The score contribution of an "exact continuous" observation $y = (\underline{y} + \bar{y})/2$ from an absolutely continuous distribution is approximated by the gradient of the log-density

$$
\begin{aligned}
\boldsymbol{s}(\boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]) \approx \frac{\partial \log(f_Y(y \mid \boldsymbol{\vartheta}))}{\partial \boldsymbol{\vartheta}} &= \frac{\partial \log(f_Z(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta}))) + \log(\boldsymbol{a}'(y)^\top \boldsymbol{\vartheta})}{\partial \boldsymbol{\vartheta}} \\
&= \boldsymbol{a}(y) \frac{f_Z'(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta})}{f_Z(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta})} + \frac{\boldsymbol{a}'(y)}{\boldsymbol{a}'(y)^\top \boldsymbol{\vartheta}}.
\end{aligned}
$$

For an interval-censored or discrete observation $\underline{y}$ and $\bar{y}$ (the constant terms $F_Z(\boldsymbol{a}(\pm\infty)^\top \boldsymbol{\vartheta}) = F_Z(\pm\infty) = 1$ or 0 vanish) the score contribution is

$$
\begin{aligned}
\boldsymbol{s}(\boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]) &= \frac{\partial \log(\mathcal{L}(\boldsymbol{a}^\top \boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]))}{\partial \boldsymbol{\vartheta}} \\
&= \frac{\partial \log(F_Z(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta}) - F_Z(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta}))}{\partial \boldsymbol{\vartheta}} \\
&= \frac{f_Z(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\bar{y}) - f_Z(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\underline{y})}{F_Z(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta}) - F_Z(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta})}.
\end{aligned}
$$

For a truncated observation, the score function is $\boldsymbol{s}(\boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]) - \boldsymbol{s}(\boldsymbol{\vartheta} \mid Y \in (y_l, y_r])$.

The `mlt()` function uses the convenience interfaces `auglag()` for augmented Lagrangian minimization (Varadhan 2015, package **alabama**) and `BB()` for spectral projected gradients implemented in the `spg()` function of package **BB** (Varadhan and Gilbert 2009, 2014) for maximising this log-likelihood. Starting values are obtained from an estimate of the unconditional distribution function $\mathbb{P}(Y \le y)$ (for example, the empirical cumulative distribution function, a Kaplan-Meier or Turnbull estimate) via a (constrained) linear regression of $z_i = F_Z^{-1}(\mathbb{P}(Y \le y_i))$ on the design matrix of the transformation model. Optionally, columns of the underlying model matrices are scaled to $[-1, 1]$ (argument `scale = TRUE` to `mlt()`) which leads to considerable faster optimisation in many cases.

4. Transformation Analysis

Based on the maximum likelihood estimator $\hat{\boldsymbol{\vartheta}}_N$ and the most likely transformation is $\hat{h}_N$, transformation models can be analysed on different scales. Plug-in estimators for the distribution and cumulative hazard functions are given by $\hat{F}_{Y,N} = F_Z \circ \boldsymbol{a}^\top \hat{\boldsymbol{\vartheta}}_N$ and $\hat{\Lambda}_{Y,N} = -\log(1 - \hat{F}_{Y,N})$. For a continuous model, the density is given by $\hat{f}_{Y,N} = f_Z \circ \boldsymbol{a}^\top \hat{\boldsymbol{\vartheta}}_N \times \boldsymbol{a}'^\top \hat{\boldsymbol{\vartheta}}_N$ and the quantile function is obtained by numerical inversion of the distribution function. For a discrete model we get $\hat{f}_{Y,N}(y_k) = F_Z(\boldsymbol{a}(y_k)^\top \hat{\boldsymbol{\vartheta}}_N) - F_Z(\boldsymbol{a}(y_{k-1})^\top \hat{\boldsymbol{\vartheta}}_N)$ (with $F_Z(\boldsymbol{a}(y_0)^\top \hat{\boldsymbol{\vartheta}}_N) := 0$ and $F_Z(\boldsymbol{a}(y_K)^\top \hat{\boldsymbol{\vartheta}}_N) := 1$). The hazard function is $\hat{\lambda}_{Y,N} = \hat{f}_{Y,N}/(1 - \hat{F}_{Y,N})$.

The `predict()` method for `mlt` objects is the main user interface for the evaluation of these functions, the type of which is selected by its `type` argument. Conceptually, all functions are evaluated on the support of Y, *i.e.* on a grid $y_1, \dots, y_K$ (arguments `q` for the vector of grid points or `K` for the number of grid point to be generated) for continuous responses for observations with explanatory variables as given in the `newdata` argument. That means the transformation function

$$\hat{h}_N(y_k \mid \boldsymbol{x}_i) = \boldsymbol{c}(y_k, \boldsymbol{x}_i)^\top \hat{\boldsymbol{\vartheta}}_N, \quad k = 1, \dots, K; i = 1, \dots, N_\text{new}$$

is evaluated for potentially large numbers K and N_new by `predict()` and is returned as a $K \times N_\text{new}$ matrix (columns correspond to observations). Because in the most general case of a conditional distribution function the transformation function is

$$\hat{h}_N(y_k \mid \boldsymbol{x}_i) = (\boldsymbol{a}_1(y_k)^\top \otimes (\boldsymbol{b}_1(\boldsymbol{x}_i)^\top, \dots, \boldsymbol{b}(\boldsymbol{x}_i)_J^\top), -\boldsymbol{b}(\boldsymbol{x}_i)_\text{shift}^\top)^\top \hat{\boldsymbol{\vartheta}}_N$$

with $\boldsymbol{A}$ being the matrix with rows $\boldsymbol{a}_1(y_k)$ for $k = 1, \dots, K$, $\boldsymbol{B}$ the matrix with rows $(-\boldsymbol{b}_1(\boldsymbol{x}_i)^\top, \dots, \boldsymbol{b}(\boldsymbol{x}_i)_J^\top)$ for $i = 1, \dots, N_\text{new}$ and $\boldsymbol{B}_\text{shift}$ the matrix with rows $\boldsymbol{b}(\boldsymbol{x}_i)^\top$ for $i = 1, \dots, N_\text{new}$, the transformation function can be simultaneously evaluated for all $k = 1, \dots, K$ and $i = 1, \dots, N_\text{new}$ as

$$(\boldsymbol{A} \otimes \boldsymbol{B} \mid \boldsymbol{1}_K \otimes \boldsymbol{B}_\text{shift})^\top \hat{\boldsymbol{\vartheta}}_N.$$

This product is in the special form of an array model (Currie *et al.* 2006) and is computed very quickly by **mlt** using the tricks described by Currie *et al.* (2006).

5. Classical Likelihood Inference

Because the problem of estimating an unknown distribution function is embedded in the maximum likelihood framework in **mlt**, the asymptotic analysis benefits from standard results on the asymptotic behaviour of maximum likelihood estimators. The contribution of an "exact continuous" observation y from an absolutely continuous distribution to the Fisher information is approximately

$$\begin{aligned}
\boldsymbol{F}(\boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]) &\approx -\frac{\partial^2 \log(f_Y(y \mid \boldsymbol{\vartheta}))}{\partial \boldsymbol{\vartheta} \partial \boldsymbol{\vartheta}^\top} \\
&= -\left(\boldsymbol{a}(y)\boldsymbol{a}(y)^\top \left\{ \frac{f_Z''(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta})}{f_Z(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta})} - \left[\frac{f_Z'(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta})}{f_Z(\boldsymbol{a}(y)^\top \boldsymbol{\vartheta})}\right]^2 \right\} - \frac{\boldsymbol{a}'(y)\boldsymbol{a}'(y)^\top}{(\boldsymbol{a}'(y)^\top \boldsymbol{\vartheta})^2}\right).
\end{aligned}$$

For a censored or discrete observation, we have the following contribution to the Fisher information

$$\begin{aligned}
\boldsymbol{F}(\boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]) &= -\frac{\partial^2 \log(\mathcal{L}(\boldsymbol{a}^\top \boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]))}{\partial \boldsymbol{\vartheta} \partial \boldsymbol{\vartheta}^\top} \\
&= -\left\{ \frac{f_Z'(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\bar{y})\boldsymbol{a}(\bar{y})^\top - f_Z'(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\underline{y})\boldsymbol{a}(\underline{y})^\top}{F_Z(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta}) - F_Z(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta})} \right. \\
&\quad - \frac{[f_Z(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\bar{y}) - f_Z(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\underline{y})]}{[F_Z(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta}) - F_Z(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta}]^2} \times \\
&\quad \left. [f_Z(\boldsymbol{a}(\bar{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\bar{y})^\top - f_Z(\boldsymbol{a}(\underline{y})^\top \boldsymbol{\vartheta})\boldsymbol{a}(\underline{y})^\top] \right\}.
\end{aligned}$$

For a truncated observation, the Fisher information is given by $\boldsymbol{F}(\boldsymbol{\vartheta} \mid Y \in (\underline{y}, \bar{y}]) - \boldsymbol{F}(\boldsymbol{\vartheta} \mid Y \in (y_l, y_r])$.

Based on these results, we can construct asymptotically valid confidence intervals and confidence bands for the conditional distribution function from confidence intervals and bands for the linear functions $\boldsymbol{a}^\top \boldsymbol{\vartheta}$.

Categorical Data Analysis: Chinese Survey (Cont'd) For the proportional odds model for happiness given age and income, we compare the score function (using the `estfun()` method from package **sandwich**, Zeileis 2004; Zeileis and Lumley 2017)

```
R> summary(estfun(polr_CHFLS_2) - (-estfun(mlt_CHFLS_2)[,c(4, 5, 1:3)]))

     R_age                R_income           Very unhappy|Not too happy
 Min.   :-9.253e-04   Min.   :-0.0245392   Min.   :0.000e+00
 1st Qu.:-1.809e-04   1st Qu.:-0.0014137   1st Qu.:0.000e+00
 Median :-4.417e-05   Median :-0.0001828   Median :0.000e+00
 Mean   :-9.248e-05   Mean   : 0.0002326   Mean   :5.110e-07
 3rd Qu.: 3.964e-05   3rd Qu.: 0.0004230   3rd Qu.:0.000e+00
 Max.   : 6.521e-04   Max.   : 0.1516610   Max.   :4.845e-06
 Not too happy|Somewhat happy Somewhat happy|Very happy
 Min.   :-6.702e-06           Min.   :-1.489e-05
 1st Qu.:-1.153e-06           1st Qu.:-5.697e-07
 Median : 0.000e+00           Median : 4.545e-07
 Mean   : 1.272e-07           Mean   : 7.905e-07
 3rd Qu.: 1.315e-06           3rd Qu.: 2.543e-06
 Max.   : 7.408e-06           Max.   : 7.092e-06
```

and the standard errors of the regression coefficients. The positive effect of income is "significant" by standard measures (the classical coefficient table was obtained using `cftest()` from package **multcomp**, Hothorn *et al.* 2008, 2017a)

```
R> cbind(polr = sqrt(diag(vcov(polr_CHFLS_2))),
+        mlt = sqrt(diag(vcov(mlt_CHFLS_2)))[c(4, 5, 1:3)])

                                    polr          mlt
R_age                       5.684041e-03 5.684460e-03
R_income                    8.509055e-05 7.099287e-05
Very unhappy|Not too happy  3.536235e-01 3.522621e-01
Not too happy|Somewhat happy 2.419643e-01 2.400507e-01
Somewhat happy|Very happy   2.372414e-01 2.354997e-01

R> cftest(polr_CHFLS_2)


	 Simultaneous Tests for General Linear Hypotheses

Fit: polr(formula = R_happy ~ R_age + R_income, data = CHFLS)

Linear Hypotheses:
                Estimate Std. Error z value Pr(>|z|)
R_age == 0    -6.279e-03  5.684e-03  -1.105  0.26930
R_income == 0  2.350e-04  8.509e-05   2.762  0.00575 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Univariate p values reported)
```

```
R> cftest(mlt_CHFLS_2, parm = names(coef(polr_CHFLS_2)))


         Simultaneous Tests for General Linear Hypotheses

Fit: mlt(model = ctm_CHFLS_2, data = CHFLS, scale = TRUE)

Linear Hypotheses:
                Estimate Std. Error z value Pr(>|z|)
R_age == 0    -6.281e-03  5.684e-03  -1.105 0.269202
R_income == 0  2.350e-04  7.099e-05   3.310 0.000932 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Univariate p values reported)
```

Survival Analysis: GBSG-2 Trial (Cont'd) For the "classical" Cox model `mlt_GBSG2_1`, the standard errors of all three methods applied (`coxph()`, `mlt()`, `flexsurvspline()`) are more or less identical

```
R> cf <- coef(coxph_GBSG2_1)
R> cbind(coxph = sqrt(diag(vcov(coxph_GBSG2_1))),
+        mlt = sqrt(diag(vcov(mlt_GBSG2_1)))[names(cf)],
+        fss = sqrt(diag(vcov(fss_GBSG2_1)))[names(cf)])

                    coxph          mlt          fss
horThyes     0.1290732771 0.1293733637 0.1290343235
age          0.0093002439 0.0093123274 0.0092899409
menostatPost 0.1834799825 0.1836804956 0.1833300012
tsize        0.0039390579 0.0039383332 0.0039372592
tgrade.L     0.1898442780 0.1898998037 0.1899068142
tgrade.Q     0.1219645723 0.1220197860 0.1220298327
pnodes       0.0074480015 0.0074150148 0.0074387203
progrec      0.0005734844 0.0005740091 0.0005553729
estrec       0.0004503673 0.0004515480 0.0004309581
```

As a consequence, the corresponding coefficient tables, here produced using `cftest()`, are also rather similar

```
R> cftest(coxph_GBSG2_1)


         Simultaneous Tests for General Linear Hypotheses

Fit: coxph(formula = fm_GBSG2, data = GBSG2, ties = "breslow")

Linear Hypotheses:
```

```
                     Estimate Std. Error z value Pr(>|z|)
horThyes == 0      -0.3462416  0.1290733  -2.683  0.00731 **
age == 0           -0.0094534  0.0093002  -1.016  0.30941
menostatPost == 0   0.2581565  0.1834800   1.407  0.15943
tsize == 0          0.0077983  0.0039391   1.980  0.04773 *
tgrade.L == 0       0.5510838  0.1898443   2.903  0.00370 **
tgrade.Q == 0      -0.2011060  0.1219646  -1.649  0.09917 .
pnodes == 0         0.0487818  0.0074480   6.550 5.77e-11 ***
progrec == 0       -0.0022175  0.0005735  -3.867  0.00011 ***
estrec == 0         0.0001978  0.0004504   0.439  0.66049
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Univariate p values reported)

R> cftest(mlt_GBSG2_1, parm = names(cf))


         Simultaneous Tests for General Linear Hypotheses

Fit: mlt(model = ctm_GBSG2_1, data = GBSG2, scale = TRUE)

Linear Hypotheses:
                     Estimate Std. Error z value Pr(>|z|)
horThyes == 0      -0.3487079  0.1293734  -2.695 0.007031 **
age == 0           -0.0099320  0.0093123  -1.067 0.286178
menostatPost == 0   0.2676427  0.1836805   1.457 0.145086
tsize == 0          0.0077660  0.0039383   1.972 0.048622 *
tgrade.L == 0       0.5602231  0.1898998   2.950 0.003177 **
tgrade.Q == 0      -0.2019424  0.1220198  -1.655 0.097925 .
pnodes == 0         0.0487620  0.0074150   6.576 4.83e-11 ***
progrec == 0       -0.0022105  0.0005740  -3.851 0.000118 ***
estrec == 0         0.0001830  0.0004515   0.405 0.685294
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Univariate p values reported)

R> cftest(fss_GBSG2_1, parm = names(cf))


         Simultaneous Tests for General Linear Hypotheses

Fit: flexsurvspline(formula = fm_GBSG2, data = GBSG2, k = 9, bknots = kn,
    scale = "hazard")

Linear Hypotheses:
                     Estimate Std. Error z value Pr(>|z|)
horThyes == 0      -0.3473363  0.1290343  -2.692  0.00711 **
```

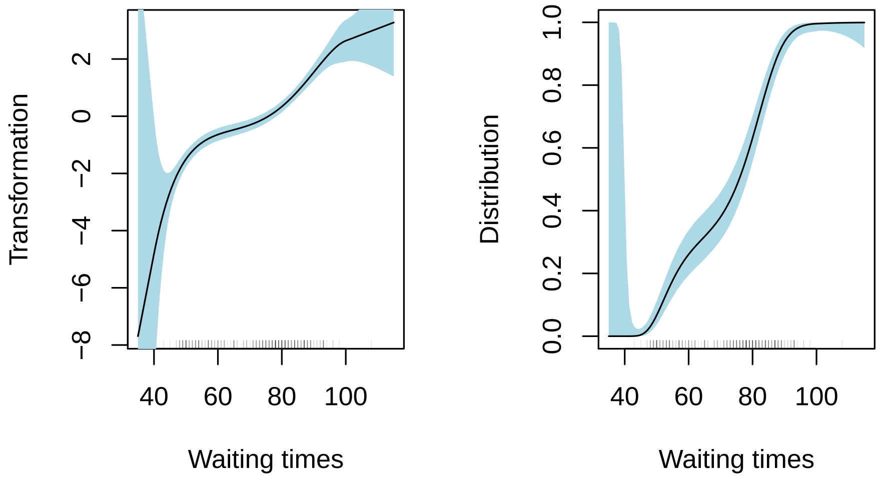

Figure 13: Old Faithful Geyser. Estimated transformation (left) and distribution functions (right) with asymptotic confidence bands.

```
age == 0           -0.0097896  0.0092899  -1.054  0.29198
menostatPost == 0   0.2672205  0.1833300   1.458  0.14495
tsize == 0          0.0079052  0.0039373   2.008  0.04467 *
tgrade.L == 0       0.5554285  0.1899068   2.925  0.00345 **
tgrade.Q == 0      -0.2078500  0.1220298  -1.703  0.08852 .
pnodes == 0         0.0485934  0.0074387   6.532 6.47e-11 ***
progrec == 0       -0.0022112  0.0005554  -3.982 6.85e-05 ***
estrec == 0         0.0001891  0.0004310   0.439  0.66076
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Univariate p values reported)
```

Density Estimation: Geyser Data (Cont'd) A relatively simple method for the construction of asymptotic confidence bands is based on the multivariate joint normal distribution of the model coefficients $\hat{\boldsymbol{\vartheta}}_N$. A confidence band for the unconditional transformation function of waiting times is derived from simultaneous confidence intervals for the linear function $\boldsymbol{A}\hat{\boldsymbol{\vartheta}}_N$ as implemented in package **multcomp**. Here $\boldsymbol{A}$ is the matrix of Bernstein basis functions evaluated on a grid of `K` waiting times $y_1, \dots, y_K$. The quantile adjusted for multiplicity is then used on a finer grid of `cheat` values for plotting purposes

```
R> cb_w <- confband(mlt_w, newdata = data.frame(1), K = 20, cheat = 100)
```

The result is shown in Figure 13 on the scale of the transformation and distribution function.

6. Simulation-based Likelihood Inference

The fully specified probabilistic model $(F_Z, \boldsymbol{a}, \hat{\boldsymbol{\vartheta}}_N)$ allows sampling from the distribution $\hat{\mathbb{P}}_Y$. For estimated parameters $\hat{\boldsymbol{\vartheta}}_N$, this model-based or "parametric" bootstrap from $\hat{\mathbb{P}}_Y$ can be implemented by the probability integral transform, *i.e.* $Z_1, \dots, Z_N \overset{\text{iid}}{\sim} \mathbb{P}_Z$ is drawn and then $Y_i^\star = \inf\{y \in \Xi \mid \boldsymbol{a}(y)^\top \hat{\boldsymbol{\vartheta}}_N \geq Z_i\}$ is determined by numerical inversion of the distribution function. The `simulation()` method for `mlt` objects first computes the distribution function over a grid of `K` response values, draws `nsim` times `nrow(newdata)` random samples from $\mathbb{P}_Z$ and returns the intervals $y_k < Y_i^\star < y_{k+1}$ (`interpolate = FALSE`) or a linear interpolation thereof (`interpolate = TRUE`).

Density Estimation: Geyser Data (Cont'd) The model-based bootstrap analogue of the confidence band for the distribution function of waiting times (Figure 13, right panel) based on 100 bootstrap samples is generated from the following code. First, 100 samples of size $N = 299$ are drawn from the fitted unconditional model `mlt_w` for waiting times. For each sample, the model `ctm_w` is refitted, using the parameters $\hat{\boldsymbol{\vartheta}}_N$ as starting values (`theta`) to speed-up the computations. Next, the log-likelihood ratio

$$\sum_{i=1}^{N} \log(\mathcal{L}(\boldsymbol{a}^\top \hat{\boldsymbol{\vartheta}}_N^\star \mid Y_i^\star) - \sum_{i=1}^{N} \log(\mathcal{L}(\boldsymbol{a}^\top \hat{\boldsymbol{\vartheta}}_N \mid Y_i^\star)$$

is computed for each sample. Last, distribution and density functions are obtained for each of the models in this small loop

```
R> new_w <- simulate(mlt_w, nsim = 100)
R> llr <- numeric(length(new_w))
R> pdist <- vector(mode = "list", length = length(new_w))
R> pdens <- vector(mode = "list", length = length(new_w))
R> ngeyser <- geyser
R> q <- mkgrid(var_w, 100)[[1]]
R> for (i in 1:length(new_w)) {
+      ngeyser$waiting <- new_w[[i]]
+      mlt_i <- mlt(ctm_w, data = ngeyser, scale = TRUE,
+                   theta = coef(mlt_w))
+      llr[[i]] <- logLik(mlt_i) - logLik(mlt_i, parm = coef(mlt_w))
+      pdist[[i]] <- predict(mlt_i, newdata = data.frame(1),
+                            type = "distribution", q = q)
+      pdens[[i]] <- predict(mlt_i, newdata = data.frame(1),
+                            type = "density", q = q)
+  }
```

The distribution and density functions corresponding to log-likelihood ratios less then the 95% quantile of the 100 log-likelihood ratios, *i.e.* after removal of five extreme curves, are plotted in Figure 14 and can be interpreted as a band around the distribution and density function. The asymptotic band for the distribution function nicely fits the band obtained from the bootstrap sample but the latter does not deteriorate for probabilities close to zero and one.

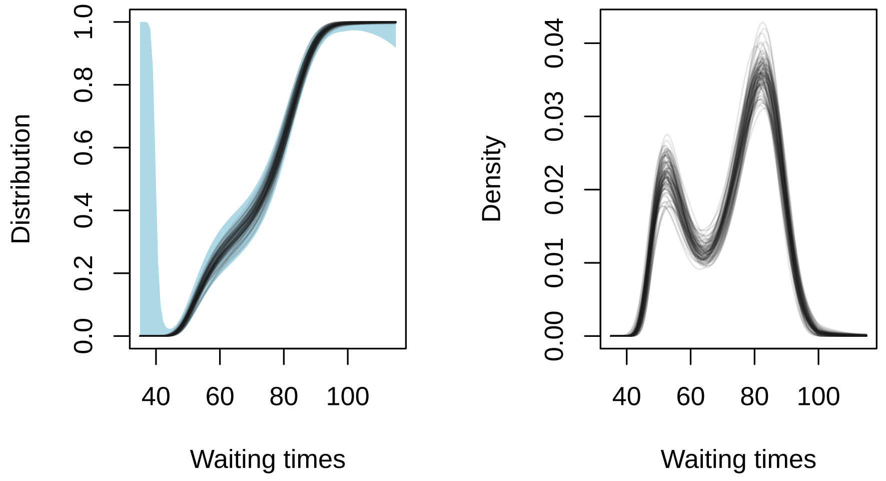

Figure 14: Old Faithful Geyser. Model-based bootstrap for distribution (left, the blue area depicts the asymptotic confidence band from Figure 13) and density (right) function for the unconditional transformation model parameterised in terms of a Bernstein polynomials of order 8.

In the conditional case, `nsim` samples from the N conditional distributions $\hat{\mathbb{P}}_{Y|\boldsymbol{X}=\boldsymbol{x}_i}, i = 1, \dots, N$ are drawn by `simulate()`.

7. Summary

The computational framework implemented in package **mlt** allows fitting of a large class of transformation models. Many established R add-on packages implement special cases, mostly in the survival analysis context. The most prominent one is the **survival** package (Therneau and Grambsch 2000; Therneau 2017) with partial likelihood estimation of the Cox model in `coxph()` and parametric linear transformation models in `survreg()`. Parametric proportional hazards (`phreg()`) and accelerated failure time models are also available from package **eha** (Broström 2017). The results obtained using these functions are practically identical to those obtained from the unified implementation of transformation models in **mlt** as shown in the various examples presented in Section 2. Other packages offer estimation of the Cox model for interval-censored responses, for example **coxinterval** (Boruvka and Cook 2015), **MIICD** (Delord 2017) and **ICsurv** (McMahan and Wang 2014). No special treatment of this situation is necessary in **mlt** as the likelihood maximised by `mlt()` allows arbitrary schemes of random censoring and also truncation.

Alternative likelihood approaches to transformation models often parameterise the hazard or log-hazard function by some spline, including the R add-on packages **polspline** (Kooperberg 2015), **logspline** (Kooperberg 2016), **bshazard** (Paola Rebora and Reilly 2014) and **gss**

(Gu 2014, 2017). Packages **muhaz** (Hess and Gentleman 2014) and **ICE** (Braun 2013) implement kernel smoothing for hazard function estimation, the latter package allows for interval-censored responses. The penalised maximum likelihood estimation procedure for simultaneous estimation of the baseline hazard function and the regression coefficients in a Cox model is available from package **survivalMPL** (Couturier *et al.* 2017). Estimation of the unconstrained log-hazard function is theoretically attractive but too erratic estimates have to be dealt with by some form of penalisation. A direct parameterisation of the transformation function, *i.e.* the log-cumulative baseline hazard in the Cox model, only requires monotone increasing functions to be fitted. Thus, penalisation is not necessary but one has to deal with a constrained problem. Package **mlt** follows the latter approach.

Based on the estimation equation procedure by Chen *et al.* (2002), **TransModel** (Zhou *et al.* 2017) implements continuous time proportional hazards and proportional odds linear transformation models. Time-varying coefficients can be estimated using packages **dynsurv** (Wang *et al.* 2017) and **timereg** (Scheike and Martinussen 2006; Scheike and Zhang 2011; Scheike *et al.* 2017) Discrete proportional odds or proportional hazards models for the analysis of ordered categorical responses are implemented in packages **MASS** (Venables and Ripley 2002; Ripley 2017), **ordinal** (Christensen 2015) and **VGAM** (Yee 2010, 2017).

Maximum likelihood estimators for a fair share of the models implemented in these established packages can be re-implemented using the computational infrastructure offered by package **mlt**. The availability of (at least) two independent implementations allows package developers to validate their implementation. In fact, some errors in earlier development versions of **mlt** could be detected by comparing model outputs. Because the implementation of maximum likelihood estimation for conditional transformation models presented in this paper relies on a rather dense code base (200 lines of pure R code in **variables**, 860 lines in **basefun** and 1450 lines in **mlt**, all convenience functions and user interfaces included), the likelihood of implementation errors is smaller compared the likelihood of errors in any of the plethora of alternative implementations of special linear transformation models (but not zero, of course).

The modelling abilities of **mlt** go beyond what is currently available in R. Maybe the practically most interesting example is distribution regression, *i.e.* transformation models with response-varying regression coefficients. The only special case available in R we are aware of are Cox models with time-varying effects in packages **dynsurv** and **timereg**. For other models, such as the distribution regression analysis of the Boston Housing data presented here, or for non-proportional odds or non-proportional hazards models, implementations are lacking. Our analysis of the bird counts example is novel also from a modelling point of view as linear transformation models for count data are still waiting to be kissed awake.

One unique feature of **mlt** is the strict separation of model specification (using `ctm()`) and model estimation (using `mlt()`) allowing computations on unfitted models. Models are specified by combinations of basis functions instead of using the rather restrictive formula language. In addition, model coefficients can be altered by the user, for example for computing conditional distribution or density functions or for the evaluation of the log-likelihood at arbitrary values of the parameters. The `simulate()` method for `mlt` objects can always be used to draw samples from fitted (or unfitted) transformation models. Thus, the implementation of parametric bootstrap procedures is a straightforward exercise.

The very flexible and powerful user interface of **mlt** will, however, be incomprehensible for most of its potential users. Because transformation models seldomly receive the attention they deserve in statistics courses, the unorthodox presentation of regression models ignoring the fences between the traditionally compartmentalised fields of “regression analysis”, “survival analysis” or “categorical data analysis” in this documentation of **mlt** will likely also confuse many statisticians, let alone data or subject-matter scientists. Future efforts will concentrate on the implementation of convenience interfaces, *i.e.* higher level user interfaces for special forms of transformation models, which resemble the traditional notational and naming conventions from the statistical modelling literature.

References

Azzalini A, Bowman AW (1990). “A Look at Some Data on the Old Faithful Geyser.” *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **39**(3), 357–365. doi:10.2307/2347385.

Bates D, Maechler M (2017). ***Matrix**: Sparse and Dense Matrix Classes and Methods.* R package version 1.2-12, URL https://CRAN.R-project.org/package=Matrix.

Boruvka A, Cook RJ (2015). ***coxinterval**: Cox-Type Models for Interval-Censored Data.* R package version 1.2, URL https://CRAN.R-project.org/package=coxinterval.

Box GEP, Cox DR (1964). “An Analysis of Transformations.” *Journal of the Royal Statistical Society B*, **26**(2), 211–252.

Braun WJ (2013). ***ICE**: Iterated Conditional Expectation.* R package version 0.69, URL https://CRAN.R-project.org/package=ICE.

Broström G (2017). ***eha**: Event History Analysis.* R package version 2.5.1, URL https://CRAN.R-project.org/package=eha.

Chen K, Jin Z, Ying Z (2002). “Semiparametric Analysis of Transformation Models with Censored Data.” *Biometrika*, **89**(3), 659–668. doi:10.1093/biomet/89.3.659.

Chernozhukov V, Fernández-Val I, Melly B (2013). “Inference on Counterfactual Distributions.” *Econometrica*, **81**(6), 2205–2268. doi:10.3982/ECTA10582.

Christensen RHB (2015). ***ordinal**: Regression Models for Ordinal Data.* R package version 2015.6-28, URL https://CRAN.R-project.org/package=ordinal.

Couturier DL, Ma J, Heritier S, Manuguerra M (2017). ***survivalMPL**: Penalised Maximum Likelihood for Survival Analysis Models.* R package version 0.2, URL https://CRAN.R-project.org/package=survivalMPL.

Croissant Y, Zeileis A (2016). ***truncreg**: Truncated Gaussian Regression Models.* R package version 0.2-4, URL https://CRAN.R-project.org/package=truncreg.

Currie ID, Durban M, Eilers PHC (2006). “Generalized Linear Array Models With Applications to Multidimensional Smoothing.” *Journal of the Royal Statistical Society B*, **68**(2), 259–280. doi:10.1111/j.1467-9868.2006.00543.x.

Curtis SM, Ghosh SK (2011). "A Variable Selection Approach to Monotonic Regression with Bernstein Polynomials." *Journal of Applied Statistics*, **38**(5), 961–976. doi:10.1080/02664761003692423.

Delord M (2017). ***MIICD**: Multiple Imputation for Interval Censored Data*. R package version 2.4, URL https://CRAN.R-project.org/package=MIICD.

Fahrmeir L, Kneib T, Lang S, Marx B (2013). *Regression. Models, Methods and Applications.* Springer-Verlag, New York, U.S.A.

Farouki RT (2012). "The Bernstein Polynomial Basis: A Centennial Retrospective." *Computer Aided Geometric Design*, **29**(6), 379–419. doi:10.1016/j.cagd.2012.03.001.

Fisher RA (1934). "Two New Properties of Mathematical Likelihood." *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, **53**, 285–306.

Foresi S, Peracchi F (1995). "The Conditional Distribution of Excess Returns: An Empirical Analysis." *Journal of the American Statistical Association*, **90**(430), 451–466. doi:10.1080/01621459.1995.10476537.

Fredriks AM, van Buuren S, Burgmeijer RJF, Meulmeester JF, Beuker RJ, Brugman E, Roede MJ, Verloove-Vanhorick SP, Wit J (2000). "Continuing Positive Secular Growth Change in The Netherlands 1955–1997." *Pediatric Research*, **47**(3), 316–323.

Gu C (2014). "Smoothing Spline ANOVA Models: R Package gss." *Journal of Statistical Software*, **58**(5), 1–25. URL http://www.jstatsoft.org/v58/i05/.

Gu C (2017). ***gss**: General Smoothing Splines*. R package version 2.1-7, URL https://CRAN.R-project.org/package=gss.

Hess K, Gentleman R (2014). ***muhaz**: Hazard Function Estimation in Survival Analysis*. R package version 1.2.6, URL https://CRAN.R-project.org/package=muhaz.

Hothorn T (2016). ***variables**: Variable Descriptions*. R package version 0.0-30, URL https://CRAN.R-project.org/package=variables.

Hothorn T (2017a). ***basefun**: Infrastructure for Computing with Basis Functions*. R package version 0.0-38, URL http://CRAN.R-project.org/package=basefun.

Hothorn T (2017b). ***mlt**: Most Likely Transformations*. R package version 0.2-1, URL http://CRAN.R-project.org/package=mlt.

Hothorn T, Bretz F, Westfall P (2008). "Simultaneous Inference in General Parametric Models." *Biometrical Journal*, **50**(3), 346–363. doi:10.1002/bimj.200810425.

Hothorn T, Bretz F, Westfall P (2017a). ***multcomp**: Simultaneous Inference in General Parametric Models*. R package version 1.4-8, URL https://CRAN.R-project.org/package=multcomp.

Hothorn T, Kneib T, Bühlmann P (2014). "Conditional Transformation Models." *Journal of the Royal Statistical Society B*, **76**(1), 3–27. doi:10.1111/rssb.12017.

Hothorn T, Möst L, Bühlmann P (2017b). "Most Likely Transformations." *Scandinavian Journal of Statistics*. doi:10.1111/sjos.12291. Accepted 2017-06-19.

Hothorn T, Möst L, Bühlmann P (2017c). "Most Likely Transformations (Extended Version)." *Technical report*, arXiv 1508.06749. URL http://arxiv.org/abs/1508.06749.

Hothorn T, Müller J, Held L, Möst L, Mysterud A (2015). "Temporal Patterns of Deer-vehicle Collisions Consistent with Deer Activity Pattern and Density Increase but not General Accident Risk." *Accident Analysis & Prevention*, **81**, 143–152. doi:10.1016/j.aap.2015.04.037.

Jackson C (2017). ***flexsurv**: Flexible Parametric Survival and Multi-State Models.* R package version 1.1, URL https://CRAN.R-project.org/package=flexsurv.

Klein JP, Moeschberger MK (2003). *Survival Analysis. Techniques for Censored and Truncated Data.* 2nd edition. Springer-Verlag, New York, U.S.A.

Koenker R, Leorato S, Peracchi F (2013). "Distributional vs. Quantile Regression." *Technical Report 300*, Centre for Economic and International Studies, University of Rome Tor Vergata, Rome, Italy. URL http://ssrn.com/abstract=2368737.

Kooperberg C (2015). ***polspline**: Polynomial Spline Routines.* R package version 1.1.12, URL https://CRAN.R-project.org/package=polspline.

Kooperberg C (2016). ***logspline**: Logspline Density Estimation Routines.* R package version 2.1.9, URL https://CRAN.R-project.org/package=logspline.

Lindsey JK (1996). *Parametric Statistical Inference.* Clarendon Press, Oxford, UK.

Lindsey JK (1999). "Some Statistical Heresies." *Journal of the Royal Statistical Society: Series D (The Statistician)*, **48**(1), 1–40.

McMahan CS, Wang L (2014). ***ICsurv**: A Package for Semiparametric Regression Analysis of Interval-censored Data.* R package version 1.0, URL https://CRAN.R-project.org/package=ICsurv.

Mroz TA (1987). "The Sensitivity of an Empirical Model of Married Women's Hours of Work to Economic and Statistical Assumptions." *Econometrica*, **55**(4), 765–799.

Müller J, Hothorn T (2004). "Maximally Selected Two-Sample Statistics as a New Tool for the Identification and Assessment of Habitat Factors with an Application to Breeding Bird Communities in Oak Forests." *European Journal of Forest Research*, **123**, 218–228. doi:10.1007/s10342-004-0035-5.

Paola Rebora AS, Reilly M (2014). ***bshazard**: Nonparametric Smoothing of the Hazard Function.* R package version 1.0, URL https://CRAN.R-project.org/package=bshazard.

Parish WL, Laumann EO (2009). "Chinese Health and Family Life Survey." Accecibilty verified on May 18, 2009, URL http://www.spc.uchicago.edu/prc/chfls.php.

Ripley B (2016). ***nnet**: Feed-Forward Neural Networks and Multinomial Log-Linear Models.* R package version 7.3-12, URL https://CRAN.R-project.org/package=nnet.

Ripley B (2017). ***MASS**: Support Functions and Datasets for Venables and Ripley's MASS.* R package version 7.3-47, URL https://CRAN.R-project.org/package=MASS.

Royston P, Parmar MKB (2002). "Flexible Parametric Proportional-hazards and Proportional-odds Models for Censored Survival Data, with Application to Prognostic Modelling and Estimation of Treatment Effects." *Statistics in Medicine*, **21**(15), 2175–2197. doi:10.1002/sim.1203.

Scheike T, Martinussen T, Silver J, Holst K (2017). ***timereg**: Flexible Regression Models for Survival Data.* R package version 1.9.1, URL https://CRAN.R-project.org/package=timereg.

Scheike TH, Martinussen T (2006). *Dynamic Regression Models for Survival Data.* Springer-Verlag, New York, U.S.A.

Scheike TH, Zhang MJ (2011). "Analyzing Competing Risk Data Using the R timereg Package." *Journal of Statistical Software*, **38**(2), 1–15. URL http://www.jstatsoft.org/v38/i02/.

Schumacher M, Basert G, Bojar H, Hübner K, Olschewski M, Sauerbrei W, Schmoor C, Beyerle C, Neumann RLA, Rauschecker, H F, for the German Breast Cancer Study Group (1994). "Randomized 2 × 2 Trial Evaluating Hormonal Treatment and the Duration of Chemotherapy in Node-positive Breast Cancer Patients." *Journal of Clinical Oncology*, **12**, 2086–2093.

Stasinopoulos DM, Rigby RA (2007). "Generalized Additive Models for Location Scale and Shape (GAMLSS) in R." *Journal of Statistical Software*, **23**(7), 1–46. URL http://www.jstatsoft.org/v23/i07.

Therneau TM (2017). ***survival**: Survival Analysis.* R package version 2.41-3, URL https://CRAN.R-project.org/package=survival.

Therneau TM, Grambsch PM (2000). *Modeling Survival Data: Extending the Cox Model.* Springer-Verlag, New York, U.S.A.

Varadhan R (2015). ***alabama**: Constrained Nonlinear Optimization.* R package version 2015.3-1, URL https://CRAN.R-project.org/package=alabama.

Varadhan R, Gilbert P (2009). "BB: An R Package for Solving a Large System of Nonlinear Equations and for Optimizing a High-Dimensional Nonlinear Objective Function." *Journal of Statistical Software*, **32**(4), 1–26. URL http://www.jstatsoft.org/v32/i04/.

Varadhan R, Gilbert P (2014). ***BB**: Solving and Optimizing Large-Scale Nonlinear Systems.* R package version 2014.10-1, URL https://CRAN.R-project.org/package=BB.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S.* Fourth edition. Springer-Verlag, New York, U.S.A. ISBN 0-387-95457-0, URL http://www.stats.ox.ac.uk/pub/MASS4.

Wang W, Chen MH, Wang X, Yan J (2017). ***dynsurv**: Dynamic Models for Survival Data.* R package version 0.3-6, URL https://CRAN.R-project.org/package=dynsurv.

Wood S (2017). ***mgcv**: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation.* R package version 1.8-22, URL https://CRAN.R-project.org/package=mgcv.

Wood SN (2006). *Generalized Additive Models: An Introduction with R.* Chapman & Hall/CRC Press, Boca Raton, Florida, U.S.A.

Yee TW (2010). "The VGAM Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34. URL http://www.jstatsoft.org/v32/i10/.

Yee TW (2017). ***VGAM**: Vector Generalized Linear and Additive Models.* R package version 1.0-4, URL https://CRAN.R-project.org/package=VGAM.

Zeileis A (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. URL http://www.jstatsoft.org/v11/i10/.

Zeileis A, Lumley T (2017). ***sandwich**: Robust Covariance Matrix Estimators.* R package version 2.4-0, URL https://CRAN.R-project.org/package=sandwich.

Zhou J, Zhang J, Lu W (2017). ***TransModel**: Fit Linear Transformation Models for Right Censored Data.* R package version 2.1, URL https://CRAN.R-project.org/package=TransModel.

Appendix

A. The variables Package

The **variables** package (Hothorn 2016) offers a small collection of classes and methods for specifying and computing on abstract variable descriptions. The main purpose is to allow querying properties of variables without having access to observations. A variable description allows to extract the name (`variable.name()`), description (`desc()`) and unit (`unit()`) of the variable along with the support (`support()`) and possible bounds (`bounds()`) of the measurements. The `mkgrid()` method generates a grid of observations from the variable description. The package differentiates between factors, ordered factors, discrete, and continuous numeric variables.

A.1. Unordered Factors

We use eye color as an example of an unordered factor. The corresponding variable description is defined by the name, description and levels of this factor

```
R> f_eye <- factor_var("eye", desc = "eye color",
+                       levels = c("blue", "brown", "green", "grey", "mixed"))
```

The properties of this factor are

```
R> variable.names(f_eye)

[1] "eye"

R> desc(f_eye)

[1] "eye color"

R> variables::unit(f_eye)

[1] NA

R> support(f_eye)

$eye
[1] blue  brown green grey  mixed
Levels: blue brown green grey mixed

R> bounds(f_eye)

$eye
[1] NA
```

```
R> is.bounded(f_eye)
```

```
[1] TRUE
```

and we can generate values, *i.e.* an instance of this factor with unique levels, via

```
R> mkgrid(f_eye)
```

```
$eye
[1] blue  brown green grey  mixed
Levels: blue brown green grey mixed
```

A.2. Ordered Factors

An ordered factor, temperature in categories is used here as an example, is defined as in the unordered case

```
R> o_temp <- ordered_var("temp", desc = "temperature",
+                        levels = c("cold", "lukewarm", "warm", "hot"))
```

and the only difference is that explicit bounds are known

```
R> variable.names(o_temp)
```

```
[1] "temp"
```

```
R> desc(o_temp)
```

```
[1] "temperature"
```

```
R> variables::unit(o_temp)
```

```
[1] NA
```

```
R> support(o_temp)
```

```
$temp
[1] cold     lukewarm warm     hot
Levels: cold < lukewarm < warm < hot
```

```
R> bounds(o_temp)
```

```
$temp
[1] cold hot
Levels: cold < lukewarm < warm < hot
```

```
R> is.bounded(o_temp)

[1] TRUE

R> mkgrid(o_temp)

$temp
[1] cold     lukewarm warm     hot
Levels: cold < lukewarm < warm < hot
```

A.3. Discrete Numeric Variables

Discrete numeric variables are defined by `numeric_var()` with integer-valued `support` argument, here using age of a patient as example

```
R> v_age <- numeric_var("age", desc = "age of patient",
+                       unit = "years", support = 25:75)
```

The variable is bounded with finite support

```
R> variable.names(v_age)

[1] "age"

R> desc(v_age)

[1] "age of patient"

R> variables::unit(v_age)

[1] "years"

R> support(v_age)

$age
 [1] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
[24] 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
[47] 71 72 73 74 75

R> bounds(v_age)

$age
[1] 25 75

R> is.bounded(v_age)
```

```
[1] TRUE
```

and the support is returned in

```
R> mkgrid(v_age)

$age
 [1] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
[24] 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
[47] 71 72 73 74 75
```

A.4. Continuous Numeric Variables

For conceptually continuous variables the `support` argument is a double vector with two elements representing an interval acting as the support for any basis function to be defined later. The variable may or may not be bounded ($\pm\infty$ is allowed). For generating equidistant grids, `support + add` is used for unbounded variables or the corresponding finite boundaries if `add` is zero. Daytime temperature at Zurich, for example, could be presented as

```
R> v_temp <- numeric_var("ztemp", desc = "Zurich daytime temperature",
+                        unit = "Celsius", support = c(-10.0, 35.0),
+                        add = c(-5, 5), bounds = c(-273.15, Inf))
```

Basis functions for this variable shall be defined for temperatures between -10 and 35 degrees Celsius

```
R> variable.names(v_temp)

[1] "ztemp"

R> desc(v_temp)

[1] "Zurich daytime temperature"

R> variables::unit(v_temp)

[1] "Celsius"

R> support(v_temp)

$ztemp
[1] -10  35

R> bounds(v_temp)

$ztemp
[1] -273.15     Inf
```

```
R> is.bounded(v_temp)

[1] TRUE
```

One might be interested in evaluating model predictions outside `support` as defined by the `add` argument, so `mkgrid()` generates a equidistant grid between −15 and 40 degrees Celsius

```
R> mkgrid(v_temp, n = 20)

$ztemp
 [1] -15.0000000 -12.1052632  -9.2105263  -6.3157895  -3.4210526
 [6]  -0.5263158   2.3684211   5.2631579   8.1578947  11.0526316
[11]  13.9473684  16.8421053  19.7368421  22.6315789  25.5263158
[16]  28.4210526  31.3157895  34.2105263  37.1052632  40.0000000
```

A.5. Multiple Variables

We can join multiple variable descriptions via `c()`

```
R> vars <- c(f_eye, o_temp, v_age, v_temp)
```

and all methods discussed above work accordingly

```
R> variable.names(vars)

    eye    temp     age   ztemp
  "eye"  "temp"   "age" "ztemp"

R> desc(vars)

                         eye                         temp
                 "eye color"                "temperature"
                         age                        ztemp
            "age of patient" "Zurich daytime temperature"

R> variables::unit(vars)

      eye      temp       age     ztemp
       NA        NA   "years" "Celsius"

R> support(vars)

$eye
[1] blue  brown green grey  mixed
Levels: blue brown green grey mixed
```

```
$temp
[1] cold     lukewarm warm     hot
Levels: cold < lukewarm < warm < hot

$age
 [1] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
[24] 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
[47] 71 72 73 74 75

$ztemp
[1] -10  35

R> bounds(vars)

$eye
[1] NA

$temp
[1] cold hot
Levels: cold < lukewarm < warm < hot

$age
[1] 25 75

$ztemp
[1] -273.15     Inf

R> is.bounded(vars)

  eye  temp   age ztemp 
 TRUE  TRUE  TRUE  TRUE 

R> mkgrid(vars, n = 20)

$eye
[1] blue  brown green grey  mixed
Levels: blue brown green grey mixed

$temp
[1] cold     lukewarm warm     hot
Levels: cold < lukewarm < warm < hot

$age
 [1] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
[24] 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
[47] 71 72 73 74 75
```

```
$ztemp
 [1] -15.0000000 -12.1052632  -9.2105263  -6.3157895  -3.4210526
 [6]  -0.5263158   2.3684211   5.2631579   8.1578947  11.0526316
[11]  13.9473684  16.8421053  19.7368421  22.6315789  25.5263158
[16]  28.4210526  31.3157895  34.2105263  37.1052632  40.0000000
```

Calling

```
R> nd <- expand.grid(mkgrid(vars))
```

generates a `data.frame` with all possible values of the variables and all combinations thereof. The generic `as.vars()` takes a data frame of observations as input and derives abstract variable descriptions using the available information. The generic function `check()` returns `TRUE` if `data` matches the abstract description

```
R> check(vars, data = nd)
```

```
[1] TRUE
```

B. The basefun Package

The **basefun** package (Hothorn 2017a) implements Bernstein, Legendre, log and polynomial basis functions. In addition, facilities for treating arbitrary model matrices as basis functions evaluated on some data are available. Basis functions can be joined column-wise using `c()` or the box product can be generated using `b()`. The definition of basis functions does not require any actual observations, only variable descriptions (see Appendix A) are necessary. Each basis offers `model.matrix()` and `predict()` methods. We illustrate how one can set-up some of these basis functions in the following sections.

B.1. Polynomial Basis

For some positive continuous variable x we want to deal with the polynomial $\alpha + \beta_1 x + \beta_3 x^3$ and first set-up a variable description (see Appendix A) and the basis function (we set-up a basis of order three ommiting the quadratic term)

```
R> xvar <- numeric_var("x", support = c(0.1, pi), bounds= c(0, Inf))
R> x <- as.data.frame(mkgrid(xvar, n = 20))
R> class(pb <- polynomial_basis(xvar, coef = c(TRUE, TRUE, FALSE, TRUE)))
```

```
[1] "polynomial_basis" "basis"            "function"
```

The basis function `pb` is a `function`, therefore we can evaluate the basis as

```
R> head(pb(x))
```

```
     (Intercept)         x x^2         x^3
[1,]           1 0.0000000   0 0.000000000
[2,]           1 0.1653470   0 0.004520524
[3,]           1 0.3306940   0 0.036164195
[4,]           1 0.4960409   0 0.122054158
[5,]           1 0.6613879   0 0.289313560
[6,]           1 0.8267349   0 0.565065547
```

or, equivalently, using the corresponding `model.matrix()` method (which is preferred)

```
R> head(model.matrix(pb, data = x))
```

```
     (Intercept)         x x^2         x^3
[1,]           1 0.0000000   0 0.000000000
[2,]           1 0.1653470   0 0.004520524
[3,]           1 0.3306940   0 0.036164195
[4,]           1 0.4960409   0 0.122054158
[5,]           1 0.6613879   0 0.289313560
[6,]           1 0.8267349   0 0.565065547
```

Evaluating the polynomial for some coefficients is done by the `predict()` method

```
R> predict(pb, newdata = x, coef = c(1, 2, 0, 1.75))
```

```
 [1]  1.000000  1.338605  1.724675  2.205677  2.829075  3.642335  4.692922
 [8]  6.028303  7.695942  9.743305 12.217857 15.167065 18.638393 22.679308
[15] 27.337274 32.659757 38.694222 45.488136 53.088963 61.544169
```

which also allows derivatives to be computed

```
R> predict(pb, newdata = x, coef = c(1, 2, 0, 1.75), deriv = c(x = 1L))
```

```
 [1]  2.000000  2.143533  2.574132  3.291797  4.296528  5.588326  7.167189
 [8]  9.033118 11.186114 13.626175 16.353303 19.367496 22.668756 26.257082
[15] 30.132473 34.294931 38.744455 43.481045 48.504701 53.815423
```

B.2. Logarithmic Basis

The monotone increasing logarithmic basis $\vartheta_1 + \vartheta_2 \log(x)$ being subject to $\vartheta_2 > 0$ is defined as

```
R> class(lb <- log_basis(xvar, ui = "increasing"))
```

```
[1] "log_basis" "basis"     "function"
```

```
R> head(X <- model.matrix(lb, data = x))
```

```
     (Intercept)      log(x)
[1,]           1 -36.0436534
[2,]           1  -1.7997091
[3,]           1  -1.1065619
[4,]           1  -0.7010968
[5,]           1  -0.4134147
[6,]           1  -0.1902712
```

The model matrix contains a `constraint` attribute

```
R> attr(X, "constraint")

$ui
     [,1] [,2]
[1,]    0    1

$ci
[1] 0
```

where the linear constraints $\boldsymbol{A}\boldsymbol{\vartheta} \geq \boldsymbol{m}$ are represented by a matrix $\boldsymbol{A}$ (`ui`) and a vector $\boldsymbol{m}$ (`ci`). For $\boldsymbol{\vartheta} = (1, 2)$ the function and its derivative can be computed as

```
R> predict(lb, newdata = x, coef = c(1, 2))

 [1] -71.0873068  -2.5994182  -1.2131238  -0.4021936   0.1731705
 [6]   0.6194576   0.9841008   1.2924021   1.5594649   1.7950310
[11]   2.0057520   2.1963724   2.3703951   2.5304805   2.6786965
[16]   2.8166822   2.9457593   3.0670085   3.1813253   3.2894598
```

```
R> predict(lb, newdata = x, coef = c(1, 2), deriv = c(x = 1L))

 [1] 9.007199e+15 1.209578e+01 6.047888e+00 4.031925e+00 3.023944e+00
 [6] 2.419155e+00 2.015963e+00 1.727968e+00 1.511972e+00 1.343975e+00
[11] 1.209578e+00 1.099616e+00 1.007981e+00 9.304443e-01 8.639840e-01
[16] 8.063850e-01 7.559860e-01 7.115162e-01 6.719875e-01 6.366198e-01
```

B.3. Bernstein Basis

A monotone increasing Bernstein polynomial $\boldsymbol{a}_{\text{Bs},3}$ can be defined and evaluated as

```
R> class(bb <- Bernstein_basis(xvar, order = 3, ui = "increasing"))

[1] "Bernstein_basis" "basis"           "function"
```

```
R> head(X <- model.matrix(bb, data = x))
```

```
        Bs1(x)      Bs2(x)      Bs3(x)       Bs4(x)
[1,] 1.0986325 -0.09863254 0.000000000 0.000000e+00
[2,] 0.9369214  0.06171364 0.001354996 9.916843e-06
[3,] 0.7892824  0.19433218 0.015949084 4.363204e-04
[4,] 0.6580299  0.29552260 0.044239941 2.207583e-03
[5,] 0.5421999  0.36817664 0.083335831 6.287617e-03
[6,] 0.4408286  0.41518604 0.130345023 1.364033e-02
```

We can check if the constraints are met for some parameters

```
R> cf <- c(1, 2, 2.5, 2.6)
R> (cnstr <- attr(X, "constraint"))

$ui
3 x 4 sparse Matrix of class "dgCMatrix"

[1,] -1  1  . .
[2,]  . -1  1 .
[3,]  .  . -1 1

$ci
[1] 0 0 0

R> all(cnstr$ui %*% cf > cnstr$ci)

[1] TRUE
```

where a `Matrix` object (from package **Matrix**, Bates and Maechler 2017) represents the linear constraints. Other possible constraints include a decreasing function (`ui = "decreasing"`), positive and negative functions (`ui = "positive"` or `ui = "negative"`), as well as cyclic or integral zero functions (`ui = "cyclic"` or `ui = "zerointegral"`). The polynomial defined by the basis functions and parameters and its first derivative can be evaluated using `predict()`

```
R> predict(bb, newdata = x, coef = cf)

 [1] 0.9013675 1.0637620 1.2189539 1.3654146 1.5032406 1.6325281 1.7533736
 [8] 1.8658735 1.9701242 2.0662220 2.1542634 2.2343447 2.3065623 2.3710127
[15] 2.4277922 2.4769972 2.5187240 2.5530692 2.5801291 2.6000000

R> predict(bb, newdata = x, coef = cf, deriv = c(x = 1))

 [1] 0.98632537 0.96518023 0.91208351 0.85956975 0.80763896 0.75629112
 [7] 0.70552625 0.65534435 0.60574540 0.55672942 0.50829640 0.46044634
[13] 0.41317925 0.36649512 0.32039395 0.27487574 0.22994050 0.18558821
[19] 0.14181889 0.09863254
```

B.4. Model Matrices

Model matrices are basis functions evaluated at some data. The **basefun** package offers an `as.basis()` method for `formula` objects which basically represents unevaluated calls to `model.matrix()` with two additional arguments (`remove_intercept` removes the intercept *after* appropriate contrasts were computed and `negative` multiplies the model matrix with -1). Note that the `data` argument does not need to be a `data.frame` with observations, a variable description is sufficient. If `data` is a data frame of observations, `scale = TRUE` scales each columns of the model matrix to the unit interval. Here is an example using the iris data

```
R> iv <- as.vars(iris)
R> fb <- as.basis(~ Species + Sepal.Length + Sepal.Width,  data = iv,
+                 remove_intercept = TRUE, negative = TRUE,
+                 contrasts.args =  list(Species = "contr.sum"))
R> class(fb)

[1] "formula_basis" "basis"         "function"

R> head(model.matrix(fb, data = iris))

  Speciesversicolor Speciesvirginica Sepal.Length Sepal.Width
1                 0                0         -5.1        -3.5
2                 0                0         -4.9        -3.0
3                 0                0         -4.7        -3.2
4                 0                0         -4.6        -3.1
5                 0                0         -5.0        -3.6
6                 0                0         -5.4        -3.9
```

B.5. Combining Bases

Two (or more) basis functions can be concatenated by simply joining the corresponding model matrices column-wise. If constraints $\boldsymbol{A}_i\boldsymbol{\vartheta}_i \geq \boldsymbol{m}_i$ are present for $i = 1, 2, \dots$ the overall constraints are given by a block-diagonal matrix $\boldsymbol{A} = \text{blockdiag}(\boldsymbol{A}_1, \boldsymbol{A}_2, \dots)$, $\boldsymbol{\vartheta} = (\boldsymbol{\vartheta}_1^\top, \boldsymbol{\vartheta}_2^\top, \dots)^\top$ and $\boldsymbol{m} = (\boldsymbol{m}_1^\top, \boldsymbol{m}_2^\top, \dots)^\top$. As an example we add a positive log-transformation to a Bernstein polynomial

```
R> class(blb <- c(bern = bb,
+                 log = log_basis(xvar, ui = "increasing",
+                                 remove_intercept = TRUE)))

[1] "cbind_bases" "bases"

R> head(X <- model.matrix(blb, data = x))

        Bs1(x)      Bs2(x)      Bs3(x)       Bs4(x)      log(x)
[1,] 1.0986325 -0.09863254 0.000000000 0.000000e+00 -36.0436534
```

```
[2,] 0.9369214  0.06171364 0.001354996 9.916843e-06  -1.7997091
[3,] 0.7892824  0.19433218 0.015949084 4.363204e-04  -1.1065619
[4,] 0.6580299  0.29552260 0.044239941 2.207583e-03  -0.7010968
[5,] 0.5421999  0.36817664 0.083335831 6.287617e-03  -0.4134147
[6,] 0.4408286  0.41518604 0.130345023 1.364033e-02  -0.1902712
```

```
R> attr(X, "constraint")
```

```
$ui
4 x 5 sparse Matrix of class "dgCMatrix"

[1,] -1  1  . . .
[2,]  . -1  1 . .
[3,]  .  . -1 1 .
[4,]  .  .  . . 1

$ci
[1] 0 0 0 0
```

The box product of two basis functions is defined by the row-wise Kronecker product of the corresponding model matrices but *not* the Kronecker product of the model matrices (which would result in a matrix with the same number of columns but squared number of rows). The following definition leads to one intercept and one deviation function

```
R> fb <- as.basis(~ g, data = factor_var("g", levels = LETTERS[1:2]))
R> class(bfb <- b(bern = bb, f = fb))
```

```
[1] "box_bases" "bases"
```

```
R> nd <- expand.grid(mkgrid(bfb, n = 10))
R> head(X <- model.matrix(bfb, data = nd))
```

```
     Bs1(x):(Intercept) Bs2(x):(Intercept) Bs3(x):(Intercept)
[1,]          1.0986325        -0.09863254         0.00000000
[2,]          0.7739072         0.20707468         0.01846902
[3,]          0.5184574         0.38073757         0.09320027
[4,]          0.3264921         0.44297161         0.20033547
[5,]          0.1889422         0.42098449         0.31266696
[6,]          0.0967384         0.34198388         0.40298705
     Bs4(x):(Intercept) Bs1(x):gB Bs2(x):gB Bs3(x):gB Bs4(x):gB
[1,]       0.0000000000         0         0         0         0
[2,]       0.0005490848         0         0         0         0
[3,]       0.0076047915         0         0         0         0
[4,]       0.0302008070         0         0         0         0
[5,]       0.0774063566         0         0         0         0
[6,]       0.1582906656         0         0         0         0
```

```
R> attr(X, "constraint")

$ui
14 x 8 sparse Matrix of class "dgCMatrix"
                             
 [1,] -1  1  . .  .  .  . .
 [2,]  . -1  1 .  .  .  . .
 [3,]  .  . -1 1  .  .  . .
 [4,]  .  .  . . -1  1  . .
 [5,]  .  .  . .  . -1  1 .
 [6,]  .  .  . .  .  . -1 1
 [7,]  1  .  . .  .  .  . .
 [8,]  .  1  . .  .  .  . .
 [9,]  .  .  1 .  .  .  . .
[10,]  .  .  . 1  .  .  . .
[11,]  .  .  . .  1  .  . .
[12,]  .  .  . .  .  1  . .
[13,]  .  .  . .  .  .  1 .
[14,]  .  .  . .  .  .  . 1

$ci
 [1]    0    0    0    0    0    0 -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf
```

If `sumconstr = TRUE`, only the sum of the two functions is expected to meet the monotonicity constraint

```
R> bfb <- b(bern = bb, f = fb, sumconstr = TRUE)
R> head(X <- model.matrix(bfb, data = nd))

     Bs1(x):(Intercept) Bs2(x):(Intercept) Bs3(x):(Intercept)
[1,]          1.0986325        -0.09863254         0.00000000
[2,]          0.7739072         0.20707468         0.01846902
[3,]          0.5184574         0.38073757         0.09320027
[4,]          0.3264921         0.44297161         0.20033547
[5,]          0.1889422         0.42098449         0.31266696
[6,]          0.0967384         0.34198388         0.40298705
     Bs4(x):(Intercept) Bs1(x):gB Bs2(x):gB Bs3(x):gB Bs4(x):gB
[1,]       0.0000000000         0         0         0         0
[2,]       0.0005490848         0         0         0         0
[3,]       0.0076047915         0         0         0         0
[4,]       0.0302008070         0         0         0         0
[5,]       0.0774063566         0         0         0         0
[6,]       0.1582906656         0         0         0         0

R> attr(X, "constraint")

$ui
6 x 8 sparse Matrix of class "dgTMatrix"
```

```
[1,] -1  1  . .  .  .  . .
[2,]  . -1  1 .  .  .  . .
[3,]  .  . -1 1  .  .  . .
[4,] -1  1  . . -1  1  . .
[5,]  . -1  1 .  . -1  1 .
[6,]  .  . -1 1  .  . -1 1

$ci
[1] 0 0 0 0 0 0
```

C. The mlt Package

C.1. Specifying Response Observations

The generic function `R()` is an extention of `Surv()` (package **survival**) to possibly truncated, categorical or integer-valued response variables with methods for objects of class `Surv`, `ordered`, `factor`, `integer` and `numeric`. The main purpose is to set-up the interval $(\underline{y}, \bar{y}]$ for the evaluation of the likelihood contribution $F_Z(h(\bar{y})) - F_Z(h(\underline{y}))$. For ordered categorical responses, $\bar{y}$ is the observed level and $\underline{y}$ the level preceding $\bar{y}$ (which is missing when the first or last level was observed)

```
R> head(R(sort(unique(CHFLS$R_happy))))

[1] (NA, Very unhappy]              (Very unhappy, Not too happy]
[3] (Not too happy, Somewhat happy] (Somewhat happy, NA]
```

For integers, $\bar{y}$ is the observation and $\underline{y} = \bar{y} - 1$

```
R> R(1:5, bounds = c(1, 5))

[1] (NA, 1] (1, 2]  (2, 3]  (3, 4]  (4, NA]
```

Numeric observations can be "exact" (meaning that the likelihood is approximated by the density evaluated at the observation), interval-censored (with left- and right-censoring as special cases) and left- or right-truncated in arbitrary combinations thereof. For example, consider ten draws from the standard normal, truncated to $(-1, 2]$ and possibly censored

```
R> x <- rnorm(10)
R> xt <- round(x[x > -1 & x <= 2], 3)
R> xl <- xt - sample(c(Inf, (0:(length(xt) - 2)) / length(xt)),
+                    replace = FALSE)
R> xr <- xt + sample(c(Inf, (0:(length(xt) - 2)) / length(xt)),
+                    replace = FALSE)
R> R(c(1.2, rep(NA, length(xt))), cleft = c(NA, xl), cright = c(NA, xr),
+    tleft = -1, tright = 2)
```

```
 [1] {1.2| (-1, 2]}
 [2] {(0.143222222222222, 1.03211111111111]| (-1, 2]}
 [3] {(-1.07566666666667, -0.409]| (-1, 2]}
 [4] {(-0.116555555555556, Inf]| (-1, 2]}
 [5] {(-0.940222222222222, -0.162444444444444]| (-1, 2]}
 [6] {(-Inf, 1.03444444444444]| (-1, 2]}
 [7] {(-0.634333333333333, 0.365666666666667]| (-1, 2]}
 [8] {(1.50488888888889, 1.94933333333333]| (-1, 2]}
 [9] {(-0.334, -0.111777777777778]| (-1, 2]}
[10] {(-0.370444444444444, 0.851777777777778]| (-1, 2]}
```

Censoring and truncation (via the `cleft`, `cright`, `tleft` and `tright` arguments) are also implemented for ordered categorical and integer responses. `Surv` objects are simply converted to the alternative format without the possibility to define extra arguments

```
R> head(geyser$duration)
```

```
[1] 4.016667        2.150000        [4.000000, Inf] [4.000000, Inf]
[5] [4.000000, Inf] [0.000000,   2]
```

```
R> head(R(geyser$duration))
```

```
[1] 4.0166667 2.15      (4, Inf]  (4, Inf]  (4, Inf]  (0, 2]
```

It should be noted that the measurement scale of the observations has nothing to do with the measurement scale of the response Y in the model and the corresponding basis functions. Discrete or continuous models are specified based on the abstract variable description (Appendix A).

C.2. Methods for Conditional Transformation Models

Methods for the generic functions `bounds()`, `variable.names()`, `model.matrix()`, `mkgrid()` are available for abstract model descriptions in `ctm` objects are returned by `ctm()`.

C.3. Methods for Fitted Transformation Models

For transformation models fitted using `mlt()`, the following methods are available: `update()` (for refitting the model), `logLik()` (the log-likelihood, optionally also as a function of parameters $\boldsymbol{\vartheta}$), `estfun()` (the scores), `coef()` and `vcov()` (model parameters and their covariance), `predict` (model predictions at various scales), `confband()` (confidence bands for the transformation and distribution function), `simulate()` (sampling from the model) as well as `print()`, `summary()` and `plot()`. In addition, `variable.names()`, `mkgrid()` and `bounds()` are also implemented. The `coef<-()` method changes the parameters in the model. One last example, the approximation of a χ^2_{20} distribution by an unconditional transformation model, shall illustrate how one interacts with `ctm` and `mlt` objects. We first define the “true” distribution

```
R> pY <- function(x) pchisq(x, df = 20)
R> dY <- function(x) dchisq(x, df = 20)
R> qY <- function(p) qchisq(p, df = 20)
```

and set-up a Bernstein polynomial for the transformation function

```
R> yvar <- numeric_var("y", support = qY(c(.001, 1 - .001)),
+                        bounds = c(0, Inf))
R> By <- Bernstein_basis(yvar, order = ord <- 15, ui = "increasing")
```

The *unfitted* model is now

```
R> mod <- ctm(By)
```

and we compute the true transformation function

```
R> h <- function(x) qnorm(pY(x))
R> x <- seq(from = support(yvar)[["y"]][1], to = support(yvar)[["y"]][2],
+            length.out = ord + 1)
```

and set the parameters of the model using

```
R> mlt::coef(mod) <- h(x)
```

We can now simulate from this *purely theoretical* model

```
R> d <- as.data.frame(mkgrid(yvar, n = 500))
R> d$grid <- d$y
R> d$y <- simulate(mod, newdata = d)
```

fit this model to the simulated data

```
R> fmod <- mlt(mod, data = d, scale = TRUE)
```

and compare the true (`mod`) and fitted (`fmod`) models by looking at the coefficients and log-likelihoods (evaluated for the estimated and true coefficients)

```
R> coef(mod)

     Bs1(y)      Bs2(y)      Bs3(y)      Bs4(y)      Bs5(y)      Bs6(y) 
-3.09023231 -2.24395775 -1.56832579 -0.99651738 -0.49547433 -0.04621458 
     Bs7(y)      Bs8(y)      Bs9(y)     Bs10(y)     Bs11(y)     Bs12(y) 
 0.36325683  0.74104644  1.09291321  1.42310925  1.73487215  2.03072963 
    Bs13(y)     Bs14(y)     Bs15(y)     Bs16(y) 
 2.31269685  2.58240889  2.84121268  3.09023231 
```

```
R> coef(fmod)

    Bs1(y)     Bs2(y)     Bs3(y)     Bs4(y)     Bs5(y)     Bs6(y) 
-3.6230749 -1.8051963 -1.8051963 -1.6474327  0.1065511  0.1065511 
    Bs7(y)     Bs8(y)     Bs9(y)    Bs10(y)    Bs11(y)    Bs12(y) 
 0.1065512  0.6354989  1.5383725  1.5383725  1.5383725  1.5383725 
   Bs13(y)    Bs14(y)    Bs15(y)    Bs16(y) 
 2.2770643  2.5996077  2.5996080  3.5630423 
```

```
R> logLik(fmod)

'log Lik.' -1601.633 (df=16)

R> logLik(fmod, parm = coef(mod))

'log Lik.' -1605.297 (df=16)
```

The corresponding density functions of the χ^2_{20} distribution, the approximating true transformation model `mod` and the estimated transformation model `fmod` are plotted in Figure 15 using the code shown on top of the plot.

```
R> ## compute true density
R> d$dtrue <- dY(d$grid)
R> d$dest <- predict(fmod, q = sort(d$grid), type = "density")
R> plot(mod, newdata = d, type = "density", col = "black",
+       xlab = "y", ylab = "Density", ylim = c(0, max(d$dest)))
R> lines(d$grid, d$dtrue, lty = 2)
R> lines(sort(d$grid), d$dest[order(d$grid)], lty = 3)
R> legend("topright", lty = 1:3, bty = "n",
+          legend = c("True", "Approximated", "Estimated"))
```

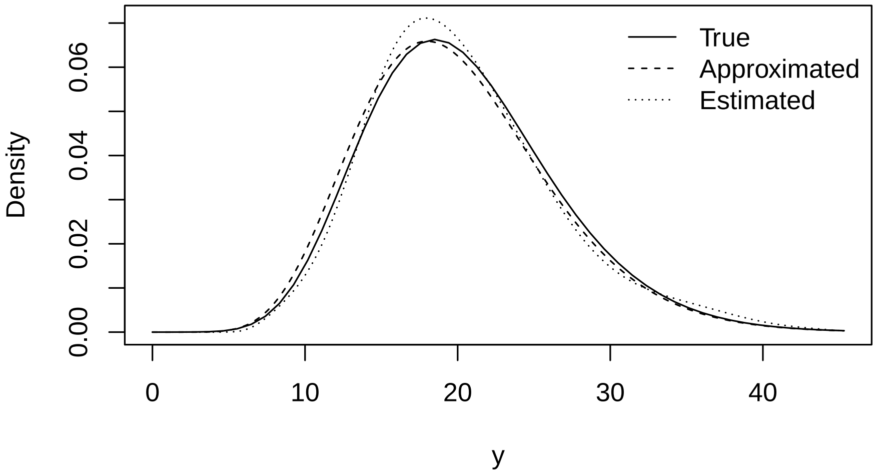

Figure 15: χ^2_{20} Data. Density of a χ^2_{20} (true), an approximating unconditional transformation model and the density fitted to a sample from the approximating model.

Affiliation:

Torsten Hothorn
Institut für Epidemiologie, Biostatistik und Prävention
Universität Zürich
Hirschengraben 84, CH-8001 Zürich, Switzerland
Torsten.Hothorn@uzh.ch
http://tiny.uzh.ch/bh