University of Zurich[UZH]

EBPI Epidemiology, Biostatistics and Prevention Institute

mlt:
Most likely transformations

Torsten Hothorn

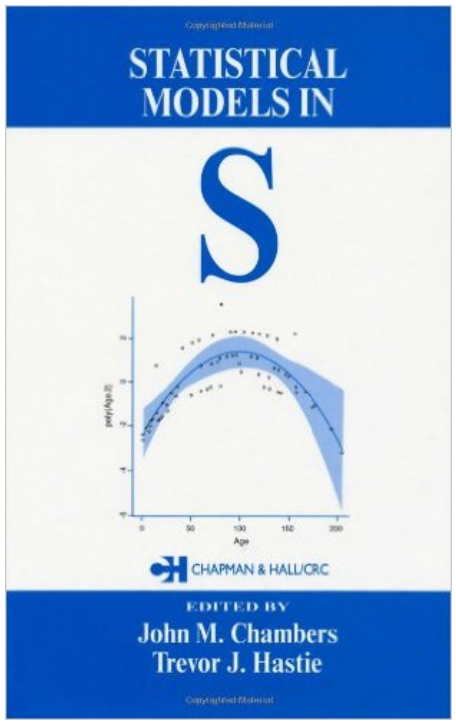
useR! 2016, 2016-06-29

Models in R

In R, (frequentist) models, data and computational routines are glued together via

```
> model(formula, data, ...)
```

following the design principles laid out in the white book.



Models in R

These ideas and their implementations still serve us well, but

- it is hard to compute on abstract (got no data yet!) probabilistic models.
  - Simulating from (unfitted) models is difficult.
  - Setting up likelihood functions to be later evaluated for yet unseen data is difficult.
- it is hard to specify more complex models in a unified way.
- it is hard analyse more complex models models in a unified way.

complex := just structuring a linear predictor won't do.

Outlook

There is some hope to make these problems less of an issue if we can unify as many interesting models as possible.

The **mlt** package allows unified specification, estimation and inference for linear models, binary GLMs (logit, probit, cloglog), accelerated failure time models, proportional hazards model (Cox), proportional odds model, discrete proportional hazards and odds models, discrete non-proportional odds and hazards models (including multinomial models), time-varying effect models (a.k.a. distribution regression), as well as for models describing proportions and counts.

How does this work?

From `lm()` to `coxph()`

Three ways to look at a normal linear model:

1.

$$\begin{aligned} Y &= \alpha + \tilde{\mathbf{x}}^\top \boldsymbol{\beta} + \sigma\varepsilon, \quad \varepsilon \sim \mathsf{N}(0, 1) \\ \mathbb{E}(Y - \alpha | \mathbf{X} = \mathbf{x}) &= \tilde{\mathbf{x}}^\top \boldsymbol{\beta} \end{aligned}$$

2.

$$\mathbb{P}(Y \leq y | \mathbf{X} = \mathbf{x}) = \Phi\left(\frac{y - \alpha - \tilde{\mathbf{x}}^\top \boldsymbol{\beta}}{\sigma}\right)$$

3.

$$\begin{aligned} \mathbb{P}(Y \leq y | \mathbf{X} = \mathbf{x}) &= \Phi(\tilde{\alpha}_1 + \tilde{\alpha}_2 y - \tilde{\mathbf{x}}^\top \tilde{\boldsymbol{\beta}}) \\ \mathbb{E}(\tilde{\alpha}_1 + \tilde{\alpha}_2 Y | \mathbf{X} = \mathbf{x}) &= \tilde{\mathbf{x}}^\top \tilde{\boldsymbol{\beta}} \end{aligned}$$

with $\tilde{\alpha}_1 = -\alpha/\sigma, \tilde{\alpha}_2 = 1/\sigma > 0$ and $\tilde{\boldsymbol{\beta}} = \boldsymbol{\beta}/\sigma$.

From `lm()` to `coxph()`

View (3) allows us to see that the normal linear model is of the form

$$\begin{aligned} \mathbb{P}(Y \leq y | \mathbf{X} = \mathbf{x}) &= F_Z(h_Y(y) - \tilde{\mathbf{x}}^\top \tilde{\boldsymbol{\beta}}) \\ \mathbb{E}(h_Y(Y) | \mathbf{X} = \mathbf{x}) &= \tilde{\mathbf{x}}^\top \tilde{\boldsymbol{\beta}} \end{aligned}$$

with F_Z a cdf of an absolutely continuous rv Z and h_Y a monotone "baseline transformation function".

With $F_Z(z) = 1 - \exp(-\exp(z))$ and "unspecified" h_Y we get the continuous proportional hazards, or Cox, model.

Other choices of F_Z and h_Y generate *all* linear transformation models.

Conditional transformation models

Similar cascades from simple to much more complex models can be understood as members of the class of models defined by

$$\mathbb{P}(Y \leq y | \mathbf{X} = \mathbf{x}) = F_Z(h(y|\mathbf{x}))$$

with conditional transformation function $h(y|\mathbf{x})$, monotone in y, and cdf F_Z.

Parameterisation: $h(y|\mathbf{x}) = \mathbf{c}(y, \mathbf{x})^\top \boldsymbol{\vartheta}$

$(F_Z, \mathbf{c}, \boldsymbol{\vartheta})$ is called "conditional transformation model" (introduced by Hothorn, Kneib, Bühlmann, 2014, JRSS-B).

Generality (1): The model

Why is this class so powerful?

With

$$\mathbb{P}(Y \leq y) = \mathbb{P}(h(Y) \leq h(y)) = F_Z(h(y))$$

we can generate *all* distributions $\mathbb{P}_Y$ from some F_Z and a corresponding h.

Suitable parameterisations of h preserve much of this generality.

Generality (2): The likelihood function

As we *always* observe intervals $(\underline{y}, \bar{y}]$ the exact likelihood is

$$\mathcal{L}(\boldsymbol{\vartheta} | Y \in (\underline{y}, \bar{y}], \mathbf{X} = \mathbf{x}) := F_Z(\mathbf{c}(\bar{y}, \mathbf{x})^\top \boldsymbol{\vartheta}) - F_Z(\mathbf{c}(\underline{y}, \mathbf{x})^\top \boldsymbol{\vartheta})$$

- Always defined, always a probability (Lindsey, 1999, JRSS-D)
- Applicable to discrete responses (e.g. `MASS::polr()`)
- Covers all types of random censoring and truncation
- For a precise datum y of some continuous Y, the likelihood can be approximated by the density

$$f_Y(y|\mathbf{x}) = f_Z(\mathbf{c}(y, \mathbf{x})^\top \boldsymbol{\vartheta}) \mathbf{c}'(y, \mathbf{x})^\top \boldsymbol{\vartheta}$$

(this is what `lm()` always does!).

Most likely transformations

Definition (Maximum likelihood estimator)

$$\hat{\boldsymbol{\vartheta}}_N := \arg\max_{\boldsymbol{\vartheta} \in \Theta} \sum_{i=1}^{N} \log(\mathcal{L}(\boldsymbol{\vartheta} | Y \in (\underline{y}, \bar{y}]_i, \mathbf{X} = \mathbf{x}_i))$$

The most likely transformation is

$$\hat{h}_N(y|\mathbf{x}) := \mathbf{c}(y, \mathbf{x})^\top \hat{\boldsymbol{\vartheta}}_N$$

with corresponding cdf

$$\hat{F}_{Y,N}(y|\mathbf{x}) := F_Z(\hat{h}_N(y|\mathbf{x})).$$

Implementation

Model specification via `mlt::ctm()`

1. Specify cdf F_Z
2. Specify variables $y, \mathbf{x}$ and basis function $\mathbf{c}$
3. (Maybe specify parameters $\boldsymbol{\vartheta} = \boldsymbol{\vartheta}_0$)

Model estimation via `mlt::mlt()`

1. Specify data
2. Estimate $\boldsymbol{\vartheta}$

Model inference via `mlt::predict()`, `mlt::simulate()`

1. Evaluate distribution, density, quantile, hazard, ... functions
2. Simulate from (unfitted or data-driven) models

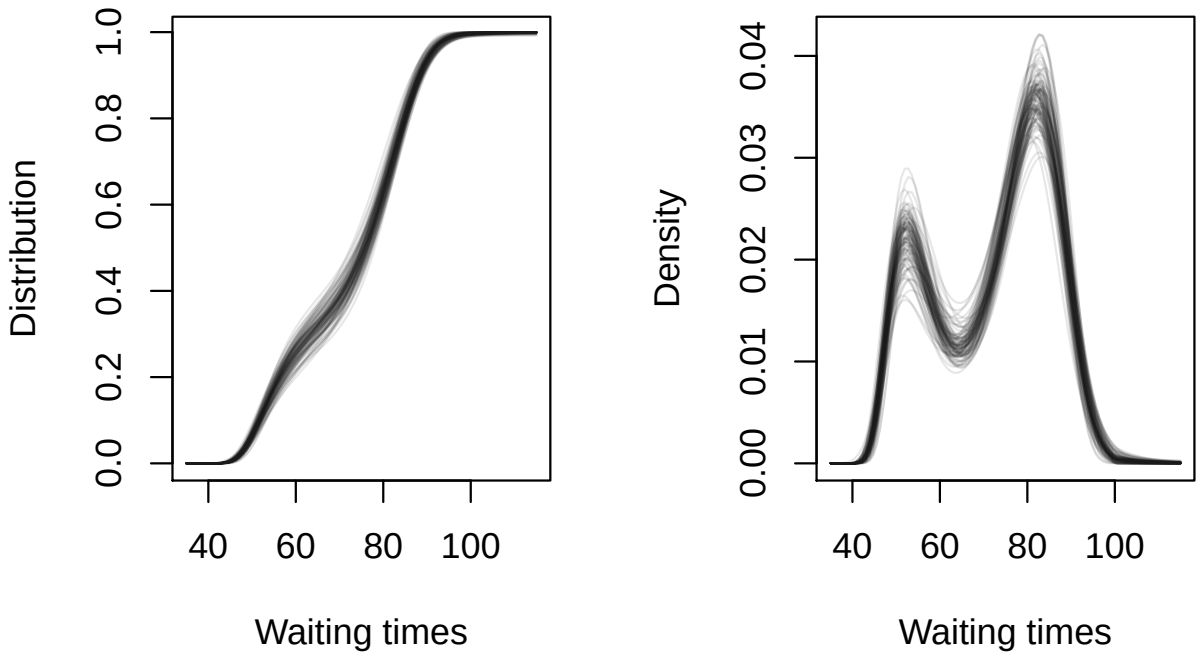
Density estimation: Old Faithful geyser

$(\Phi, \mathbf{a}_{\text{Bs},8}(\text{waiting}), \boldsymbol{\vartheta})$ with $\mathbf{a}_{\text{Bs},8}$ a Bernstein polynomial of degree 8

```
> library("mlt")
> var_w <- numeric_var("waiting",
+     support = c(40.0, 100), add = c(-5, 15),
+     bounds = c(0, Inf))
> B_w <- Bernstein_basis(var = var_w, order = 8,
+                        ui = "increasing")
> ctm_w <- ctm(B_w, todistr = "Normal")
> mlt_w <- mlt(ctm_w, data = geyser)
```

Variability assessment: Simulate and re-fit (parametric bootstrap)

Old Faithful geyser



Computing with basis functions

The whole model complexity lies in the definition of the basis function **c**. Package **basefun** offers implementations of

- polynomial,
- logarithmic,
- Bernstein, and
- "model matrix"

basis functions. Bases can be combined columns-wise using `c()` or by row-wise tensor (or box) products `b()`.

basefun has `model.matrix()` and `predict()` methods, also for derivatives of basis functions.

Model specification via `ctm()`

```
> ctm(response, interacting = NULL, shifting = NULL,
+     todistr = c("Normal", "Logistic", "MinExtrVal"),
+     ...)
```

$$\mathbf{c}(y|\mathbf{x}) = (\underbrace{\mathbf{a}(y)^\top}_{\texttt{response}} \otimes \underbrace{(\mathbf{b}_1(\mathbf{x})^\top, \ldots, \mathbf{b}(\mathbf{x})_J^\top)}_{\texttt{interacting}}, \underbrace{-\mathbf{b}(\mathbf{x})_{\text{shift}}^\top}_{\texttt{shifting}})^\top$$

Unconditional: `interacting = NULL` and `shifting = NULL`

Linear transformation model: `interacting = NULL`

Else: stratified linear transformation model, distribution regression, conditional transformation model

Survival analysis: GBSG-2

- 686 node-positive breast cancer patients (246 with and 440 without hormonal therapy), survival time y
- $F_{\text{MEV}}(z) = 1 - \exp(-\exp(z))$ minimum extreme value distribution
- Linear transformation model

$$(F_{\text{MEV}}, (\mathbf{a}(y)^\top, \mathbb{1}(\text{hormonal therapy}))^\top, \boldsymbol{\vartheta})$$

- Conditional transformation function

$$h(y|\text{treatment}) = \mathbf{a}(y)^\top \boldsymbol{\vartheta}_1 + \mathbb{1}(\text{hormonal therapy})\beta$$

- $\mathbf{a}(y)^\top \boldsymbol{\vartheta}_1$ is log-cumulative baseline hazard function
- $\beta \in \mathbb{R}$ log-hazard ratio of hormonal therapy

GBSG-2: Computing with models

Accelerated failure time models use basis

$$\mathbf{a}(y) = (1, \log(y))$$

```
> GBSG2y <- numeric_var("y", support = c(100.0, 2659),
+                       bounds = c(0, Inf))
> v_horTh <- factor_var("horTh",
+                       levels = c("no", "yes"))
> B_y <- log_basis(GBSG2y, ui = "increasing")
> B_x <- as.basis(~ horTh, data = v_horTh,
+                 remove_intercept = TRUE)
> ctm_y <- ctm(B_y, shifting = B_x,
+              todistr = "MinExtrVal")
```

GBSG-2: Computing with models

An exponential AFT with no treatment effect:

```
> coef(ctm_y)
```

```
(Intercept)      log(y)    horThyes
         NA          NA          NA
```

```
> coef(ctm_y) <- c(-7, 1, 0)
> coef(ctm_y)
```

```
(Intercept)      log(y)    horThyes
         -7           1           0
```

```
> mlt_y <- mlt(ctm_y, data = GBSG2, dofit = FALSE)
> logLik(mlt_y, parm = coef(ctm_y))
```

```
'log Lik.' -2796.426 (df=NULL)
```

GBSG-2: Computing with models

Simulate from artificial model

```
> tmp <- GBSG2
> sim_y <- simulate(ctm_y, nsim = 100, newdata = GBSG2)
> pboot <- do.call("rbind",
+     lapply(sim_y, function(sy) {
+         tmp$y <- sy
+         coef(mlt(ctm_y, data = tmp))
+     })
+ )
> colMeans(pboot)
```

```
 (Intercept)        log(y)     horThyes
-7.011955103  1.002058199  0.004382162
```

GBSG-2: Computing with models

Fit Weibull AFT to GBSG2 data

```
> mlt_y <- mlt(ctm_y, data = GBSG2, dofit = TRUE)
> coef(mlt_y)
```

```
(Intercept)      log(y)    horThyes
 -9.7791859   1.2853062  -0.3932403
```

```
> logLik(mlt_y)
```

```
'log Lik.' -2632.096 (df=3)
```

```
> sqrt(diag(vcov(mlt_y)))
```

```
(Intercept)      log(y)    horThyes
 0.46632811  0.06387437  0.12482667
```

GBSG-2: Cox model

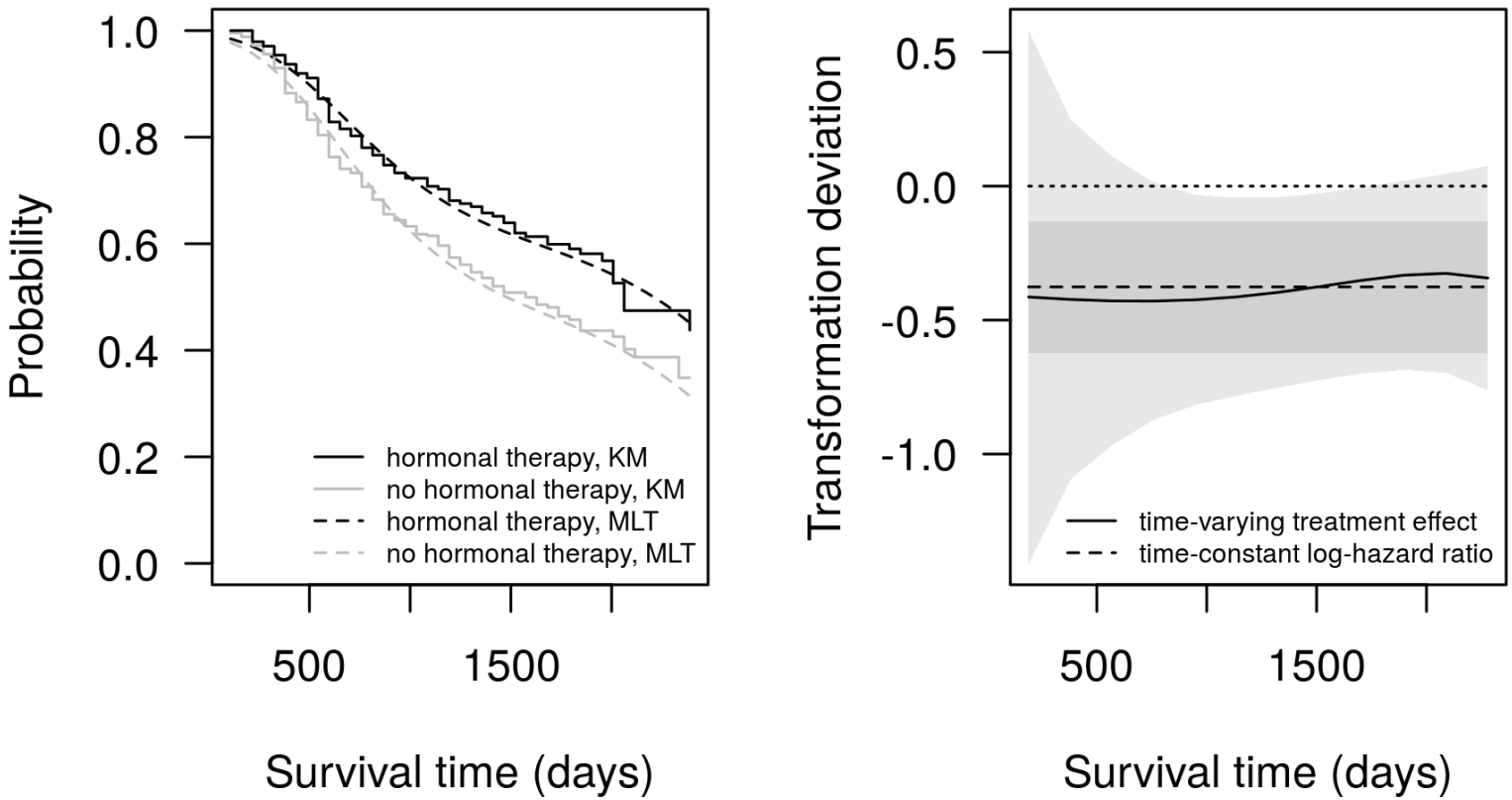
Fully parameterised Cox models with Bernstein polynomial basis functions

$$\mathbf{a}(y) = \mathbf{a}_{\text{Bs},10}(y) \text{ or } \mathbf{a}(y) = (\log(y), \mathbf{a}_{\text{Bs},10}(y))$$

```
> B_y <- Bernstein_basis(GBSG2y, ui = "increasing",
+                        order = 10)
> ctm(B_y, shifting = B_x, todistr = "MinExtrVal")
```

GBSG-2: Time-varying effects

```
> ctm(B_y, interacting = as.basis(v_horTh),
+     todistr = "MinExtrVal")
```



GBSG-2: Distribution regression

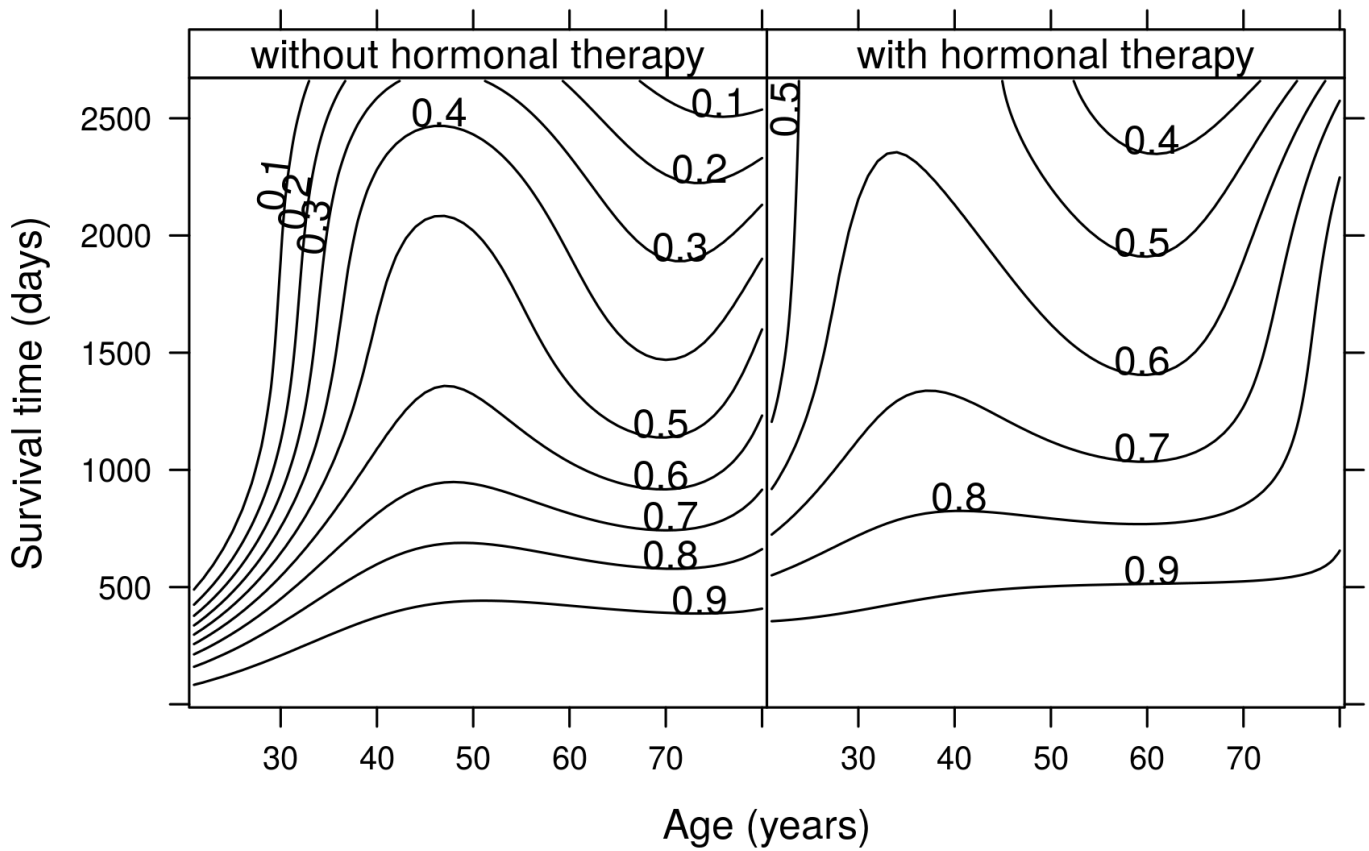
- What are the prognostic and predictive properties of age?
- Conditional transformation model

$$
\begin{aligned}
&(F_{\text{MEV}}, \\
&(\mathbf{a}_{\text{Bs},10}(y)^\top \otimes (\mathbb{1}(\text{h. therapy}), \\
&\qquad\qquad 1 - \mathbb{1}(\text{h. therapy})) \otimes \mathbf{b}_{\text{Bs},10}(\text{age})^\top)^\top, \\
&\boldsymbol{\vartheta})
\end{aligned}
$$

- Allows treatment-specific transformation functions given age
- For each treatment, survivor function varies smoothly with age

GBSG-2: Distribution regression

```
> B_age <- Bernstein_basis(numeric_var("age", ...), 3)
> ctm(B_y, interacting = b(age = B_age,
+                          horTh = as.basis(v_horTh)),
+     )
```



Quality assurance

The relevant R code in **mlt** for fitting *all* conditional transformation models is less than 1000 lines long.

Quality assurance for **mlt** was implemented via comparison to implementations of (stratified) linear transformation models under censoring and truncation in packages **stats**, **survival**, **eha**, **flexsurv**, **intcox**, **coxinterval**, **ICsurv**, **truncreg**, **MASS**, and **nnet**.

Saying that these 1000 "smart" lines of R code can replace all above packages is an exaggeration, but a modest one.

Summary

mlt allow conditional transformation models of varying complexity to be fitted and analysed in the classical maximum likelihood framework, also to randomly censored and truncated observations.

- Framework contains many important models
- Straightforward unified implementation using a standard optimiser (`BB::spg()`)
- Extremely easy to "invent" new models
- Parametric bootstrap straighforward
- Teaching: Distributions and not (just) means, connections between models

Resources

- http://CRAN.R-project.org/package=mlt
- Vignette `mlt` in http://CRAN.R-project.org/package=mlt.docreg
- Conditional Transformation Models, DOI:10.1111/rssb.12017
- Most Likely Transformations, arXiv:1508.06749