Package ‘tram’

February 25, 2025

Title Transformation Models

Version 1.2-1

Date 2025-02-25

Description Formula-based user-interfaces to specific transformation models
implemented in package 'mlt' (<DOI:10.32614/CRAN.package.mlt>, <DOI:10.32614/CRAN.package.mlt.docreg>).
Available models include Cox models, some parametric
survival models (Weibull, etc.), models for ordered categorical variables,
normal and non-normal (Box-Cox type) linear models, and continuous outcome logistic regression
(Lohse et al., 2017, <DOI:10.12688/f1000research.12934.1>). The underlying theory
is described in Hothorn et al. (2018) <DOI:10.1111/sjos.12291>. An extension to
transformation models for clustered data is provided (Barbanti and Hothorn, 2022,
<DOI:10.1093/biostatistics/kxac048>). Multivariate conditional transformation models
(Klein et al, 2022, <DOI:10.1111/sjos.12501>) and shift-scale transformation models (Siegfried et al, 2023,
<DOI:10.1080/00031305.2023.2203177>) can be fitted as well. The package contains an implementation of
a doubly robust score test, described in Kook et al. (2024, <DOI:10.1080/01621459.2024.2395588>).

Depends R (>= 3.5.0), mlt (>= 1.6-0), mvtnorm (>= 1.3-2)

Imports Formula, multcomp, variables (>= 1.0-4), basefun (>= 1.1-2),
sandwich, stats, survival, graphics, Matrix, methods

Suggests MASS, TH.data, trtf (>= 0.3-3), mlbench, knitr, quantreg,
colorspace, ATR, lme4, merDeriv, SparseGrid, alabama, numDeriv,
gridExtra, lattice, latticeExtra, HSAUR3, ordinalCont, coxme,
mlt.docreg, ordinal, coin, asht, gamlss, randomForestSRC,
tramME, glmmTMB, geepack, ranger, eha, flexsurv, frailtyEM,
frailtypack, gamlss.cens, icenReg, mpr, rms, rstpm2, timereg,
Stat2Data, cotram, latex2exp, tramvs, AER, KONPsurv,
gamlss.data

VignetteBuilder knitr

URL `http://ctm.R-forge.R-project.org`

Encoding UTF-8

NeedsCompilation no

Author Torsten Hothorn [aut, cre] (<https://orcid.org/0000-0001-8301-0471>),
Luisa Barbanti [aut] (<https://orcid.org/0000-0001-5352-5802>),
Sandra Siegfried [aut] (<https://orcid.org/0000-0002-7312-1001>),
Lucas Kook [aut] (<https://orcid.org/0000-0002-7546-7356>),
Brian Ripley [ctb],
Bill Venables [ctb],
Douglas M. Bates [ctb],
Nadja Klein [ctb]

Maintainer Torsten Hothorn <`Torsten.Hothorn@R-project.org`>

Repository CRAN

Date/Publication 2025-02-25 13:30:02 UTC

Contents

`Aareg` *Aalen Additive Hazards Model*

Description

Aalen model with fully parameterised hazard function

Usage

```
Aareg(formula, data, subset, weights, offset, cluster, na.action = na.omit, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set to `na.omit`.
`...`	additional arguments to `tram`.

Details

This function allows simultaneous estimation of the cumulative hazard parameterised by a Bernstein polynomial. The model is typically fitted with time-varying coefficients, all types of random censoring and trunction are allowed.

The responses is bounded (`bounds = c(0, Inf)`) when specified as a `Surv` object. Otherwise, bounds can be specified via `...`.

Value

An object of class `Aareg`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
data("GBSG2", package = "TH.data")
library("survival")
GBSG2$time <- as.numeric(GBSG2$time)
GBSG2$y <- with(GBSG2, Surv(time, cens))

### Cox proportional hazards model
m1 <- Coxph(y ~ horTh, data = GBSG2, support = c(1, 1500))
logLik(m1)
```

```
### Aalen additive hazards model with time-varying effects
m2 <- Aareg(y | horTh ~ 1, data = GBSG2, support = c(1, 1500))
logLik(m2)

### compare the hazard functions
nd <- data.frame(horTh = unique(GBSG2$horTh))
col <- 1:2
lty <- 1:2
plot(as.mlt(m1), newdata = nd, type = "hazard",
     col = col, lty = lty[1], xlab = "time")
plot(as.mlt(m2), newdata = nd, type = "hazard",
     col = col, lty = 2, add = TRUE)
legend("topright", col = rep(col, each = 2),
       lty = rep(1:2), bty = "n",
       legend = paste(rep(paste("horTh:",
                               levels(nd$horTh)), each = 2),
                      rep(c("Cox", "Aalen"), 2)))
```

`BoxCox` *(Similar to) Box-Cox Models*

Description

Non-normal linear regression inspired by Box-Cox models

Usage

```
BoxCox(formula, data, subset, weights, offset, cluster, na.action = na.omit, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.

`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set to `na.omit`.
`...`	additional arguments to `tram`.

Details

A normal model for transformed responses, where the transformation is estimated from the data simultaneously with the regression coefficients. This is similar to a Box-Cox transformation, but the technical details differ. Examples can be found in the package vignette.

The model is defined with a negative shift term. Large values of the linear predictor correspond to large values of the conditional expectation response (but this relationship is potentially nonlinear).

Value

An object of class `BoxCox`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
data("BostonHousing2", package = "mlbench")

lm(cmedv ~ crim + zn + indus + chas + nox + rm + age + dis +
           rad + tax + ptratio + b + lstat, data = BostonHousing2)

BoxCox(cmedv ~ chas + crim + zn + indus + nox +
               rm + age + dis + rad + tax + ptratio + b + lstat,
               data = BostonHousing2)
```

`Colr` *Continuous Outcome Logistic Regression*

Description

A proportional-odds model for continuous variables

Usage

```
Colr(formula, data, subset, weights, offset, cluster, na.action = na.omit, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset.
`...`	additional arguments to `tram`.

Details

Simultaneous estimation of all possible binary logistic models obtained by dichotomisation of a continuous response. The regression coefficients can be constant allowing for an interpretation as log-odds ratios.

The model is defined with a positive shift term, thus `exp(coef())` is the multiplicative change of the odds ratio (conditional odds of treatment or for a one unit increase in a numeric variable divided by conditional odds of reference). Large values of the linear predictor correspond to small values of the conditional expectation response (but this relationship is nonlinear).

Value

An object of class `Colr`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Tina Lohse, Sabine Rohrmann, David Faeh and Torsten Hothorn (2017), Continuous Outcome Logistic Regression for Analyzing Body Mass Index Distributions, *F1000Research*, **6**(1933), doi:10.12688/f1000research.12934.1.

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
data("BostonHousing2", package = "mlbench")

lm(cmedv ~ crim + zn + indus + chas + nox + rm + age + dis +
           rad + tax + ptratio + b + lstat, data = BostonHousing2)

Colr(cmedv ~ chas + crim + zn + indus + nox +
             rm + age + dis + rad + tax + ptratio + b + lstat,
             data = BostonHousing2)
```

`Compris` *Competing Risk Regression*

Description

An alternative approach to competing risk regression via multivariate transformation models

Usage

```
Compris(formula, data, subset, weights, na.action, offset,
        primary = c("Coxph", "Colr", "BoxCox"),
        competing = switch(primary, Coxph = "weibull",
                                    Colr = "loglogistic",
                                    BoxCox = "lognormal"),
        NPlogLik = FALSE, theta = NULL,
        optim = mltoptim(auglag = list(maxtry = 5)),
        args = list(seed = 1, type = c("MC", "ghalton"), M = 1000),
        fit = c("jointML", "none"),
        scale = FALSE, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under Details and in the package vignette. The left-hand side must be a `Surv` object, where `"event"` is specified by a factor that has levels indicating the independent censoring event, the primary event of interest and then the competing events (in this order).
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of case weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set to `na.omit`.

`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`primary`	a character defining the marginal model for the primary event of interest, that is, the first status level.
`competing`	a character defining the marginal models for the remaining competing events.
`NPlogLik`	logical, optimise nonparametric likelihood defined in terms of multivariate probabilities.
`theta`	optional starting values.
`optim`	see `Mmlt`.
`args`	a list of arguments for `lpmvnorm`.
`fit`	character vector describing how to fit the model. The default is joint likelihood estimation of all parameters.
`scale`	logical defining if variables in the linear predictor shall be scaled. Scaling is internally used for model estimation, rescaled coefficients are reported in model output.
`...`	addition arguments passed to primary or competing model function.

Details

This is a highly experimental approach to an alternative competing risk regression framework described by Czado and Van Keilegom (2023) and Deresa and Van Keilegom (2023).

Value

An object of class `Mmlt`, allowing to derive marginal time-to-event distributions for the primary event of interest and all competing events.

References

Claudia Czado and Ingrid Van Keilegom (2023). Dependent Censoring Based on Parametric Copulas. *Biometrika*, **110**(3), 721–738, doi:10.1093/biomet/asac067.

Negera Wakgari Deresa and Ingrid Van Keilegom (2023). Copula Based Cox Proportional Hazards Models for Dependent Censoring. *Journal of the American Statistical Association*, **119**(546), 1044–1054, doi:10.1080/01621459.2022.2161387.

Examples

```
if (require("randomForestSRC")) {
  library("survival")

  ## Competing risk data set involving follicular cell lymphoma
  ##   (from doi:10.1002/9780470870709)
  data("follic", package = "randomForestSRC")

  ## Therapy:
  ### Radiotherapy alone (RT) or Chemotherapy + Radiotherapy (CMTRT)
```

```
follic$ch <- factor(as.character(follic$ch),
  levels = c("N", "Y"), labels = c("RT", "CMTRT"))

## Clinical state
follic$clinstg <- factor(follic$clinstg,
  levels = 2:1, labels = c("II", "I"))

## Pre-processing as in Deresa & Van Keilegom (2023)
follic$time <- round(follic$time, digits = 3)
follic$age <- with(follic, (age - mean(age)) / sd(age)) ## standardised
follic$hgb <- with(follic, (hgb - mean(hgb)) / sd(hgb)) ## standardised

## Setup `Surv' object for fitting Compris():
### "status" indicator with levels:
###   (1) independent censoring (admin_cens)
###   (2) primary event of interest (relapse)
###   (3) dependent censoring (death)
follic$status <- factor(follic$status,
  levels = 0:2, labels = c("admin_cens", "relapse", "death"))

follic$y <- with(follic, Surv(time = time, event = status))

## Fit a Gaussian Copula-based Cox Proportional Hazards Model with
##   a marginal "Coxph" model for the primary event of interest and
##   a Weibull "Survreg" model for dependent censoring
## Use very informative starting values to keep CRAN happy
cf <- c(
        "Event_relapse.Event_relapse.Bs1(Event_relapse)" = -1.89058,
        "Event_relapse.Event_relapse.Bs2(Event_relapse)" = -1.6566,
        "Event_relapse.Event_relapse.Bs3(Event_relapse)" = -0.50329,
        "Event_relapse.Event_relapse.Bs4(Event_relapse)" = -0.50329,
        "Event_relapse.Event_relapse.Bs5(Event_relapse)" = -0.07402,
        "Event_relapse.Event_relapse.Bs6(Event_relapse)" = 0.53156,
        "Event_relapse.Event_relapse.Bs7(Event_relapse)" = 0.67391,
        "Event_relapse.Event_relapse.chCMTRT" = -0.2861,
        "Event_relapse.Event_relapse.age" = 0.43178,
        "Event_relapse.Event_relapse.hgb" = 0.02913,
        "Event_relapse.Event_relapse.clinstgI" = -0.55601,
        "Event_death.Event_death.(Intercept)" = -2.20056,
        "Event_death.Event_death.log(Event_death)" = 0.98102,
        "Event_death.Event_death.chCMTRT" = 0.25012,
        "Event_death.Event_death.age" = -0.64826,
        "Event_death.Event_death.hgb" = -0.02312,
        "Event_death.Event_death.clinstgI" = 0.57684,
        "Event_death.Event_relapse.(Intercept)" = -3.48595
       )
### gave up after multiple submissions to CRAN resulting
### in 5.02 > 5 secs

m <- Compris(y ~ ch + age + hgb + clinstg, data = follic, log_first = TRUE,
             ### arguments below speed-up example, don't use!
             theta = cf,  ### informativ starting values
             optim = mltoptim(),    ### no hessian
```

```
                args  = list(type = "ghalton",
                             M = 80)) ### only 80 MC integration points

    ### log-likelihood
    logLik(m)

    ## Similar to Table 4 of Deresa & Van Keilegom (2023),
    ## but using a Gaussian copula instead of a Gumbel copula.
    ## marginal parameters
    coef(m, type = "marginal")
    ## Kendall's tau
    coef(m, type = "Kendall")


  }
```

`Coxph` *Cox Proportional Hazards Model*

Description

Cox model with fully parameterised baseline hazard function

Usage

```
Coxph(formula, data, subset, weights, offset, cluster, na.action = na.omit, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set to `na.omit`.
`...`	additional arguments to `tram`.

Details

The original implementation of Cox models via the partial likelihood, treating the baseline hazard function as a nuisance parameter, is available in `coxph`. This function allows simultaneous estimation of the log-hazard ratios and the log-cumulative baseline hazard, the latter parameterised by a Bernstein polynomial. The model can be fitted under stratification (time-varying coefficients), all types of random censoring and trunction. An early reference to this parameterisation is McLain and Ghosh (2013).

The response is bounded (`bounds = c(0, Inf)`) when specified as a `Surv` object. Otherwise, `bounds` can be specified via `...`.

Parameters are log-hazard ratios comparing treatment (or a one unit increase in a numeric variable) with a reference.

Value

An object of class `Coxph`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Alexander C. McLain and Sujit K. Ghosh (2013). Efficient Sieve Maximum Likelihood Estimation of Time-Transformation Models, *Journal of Statistical Theory and Practice*, **7**(2), 285–303, doi:10.1080/15598608.2013.772835.

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
  data("GBSG2", package = "TH.data")

  library("survival")
  (m1 <- coxph(Surv(time, cens) ~ horTh, data = GBSG2))

  (m2 <- Coxph(Surv(time, cens) ~ horTh, data = GBSG2))

  ### McLain & Ghosh (2013); takes too long on Windows
  ## Not run: m3 <- Coxph(Surv(time, cens) ~ horTh, data = GBSG2,
               frailty = "Gamma")
## End(Not run)

  ### Wald intervals
  confint(m1)
  confint(m2)
  ### profile likelihood interval
  confint(profile(m2))
  ### score interval
  confint(score_test(m2))
  ### permutation score interval; uses permutation distribution
  ### see coin::independence_test; takes too long on Windows
  ## Not run: confint(perm_test(m2))
```

`Lehmann` *Proportional Reverse Time Hazards Linear Regression*

Description

Non-normal linear regression for Lehmann-alternatives

Usage

```
Lehmann(formula, data, subset, weights, offset, cluster, na.action = na.omit, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set to `na.omit`.
`...`	additional arguments to `tram`.

Details

This transformation model uses the cumulative distribution function for the standard Gumbel maximum extreme value distribution to map the shifted transformation function into probabilities. The exponential of the shift paramater can be interpreted as a Lehmann-alternative or reverse time hazard ratio.

Value

An object of class `Lehmann`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Erich L. Lehmann (1953), The Power of Rank Tests, *The Annals of Mathematical Statistics*, **24**(1), 23-43.

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
data("BostonHousing2", package = "mlbench")

lm(cmedv ~ crim + zn + indus + chas + nox + rm + age + dis +
           rad + tax + ptratio + b + lstat, data = BostonHousing2)

Lehmann(cmedv ~ chas + crim + zn + indus + nox +
                rm + age + dis + rad + tax + ptratio + b + lstat,
                data = BostonHousing2)
```

`Lm` *Normal Linear Model*

Description

Normal linear model with benefits

Usage

```
Lm(formula, data, subset, weights, offset, cluster, na.action = na.omit, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set to `na.omit`.
`...`	additional arguments to `tram`.

Details

A normal linear model with simulaneous estimation of regression coefficients and scale parameter(s). This function also allows for stratum-specific intercepts and variances as well as censoring and truncation in the response.

Note that the scale of the parameters is different from what is reported by `lm`; the discrepancies are explained in the package vignette.

The model is defined with a negative shift term. Large values of the linear predictor correspond to large values of the conditional expectation response.

Value

An object of class `Lm`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
data("BostonHousing2", package = "mlbench")

lm(cmedv ~ crim + zn + indus + chas + nox + rm + age + dis +
           rad + tax + ptratio + b + lstat, data = BostonHousing2)

Lm(cmedv ~ chas + crim + zn + indus + nox +
           rm + age + dis + rad + tax + ptratio + b + lstat,
           data = BostonHousing2)
```

`Mmlt` *Multivariate Conditional Transformation Models*

Description

Conditional transformation models for multivariate continuous, discrete, or a mix of continuous and discrete outcomes

Usage

```
Mmlt(..., formula = ~ 1, data, conditional = FALSE, theta = NULL, fixed = NULL,
     scale = FALSE, optim = mltoptim(auglag = list(maxtry = 5)),
     args = list(seed = 1, type = c("MC", "ghalton"), M = 1000),
     fit = c("jointML", "pseudo", "ACS", "sequential", "none"),
            ACSiter = 2)
```

Arguments

`...`	marginal transformation models, one for each response, for `Mmlt`. Additional arguments for the methods.
`formula`	a model formula describing a model for the dependency structure via the lambda parameters. The default is set to `~ 1` for constant lambdas.
`data`	a data.frame.
`conditional`	logical; parameters are defined conditionally (only possible when all models are probit models). This is the default as described by Klein et al. (2022). If `FALSE`, parameters can be directly interpreted marginally, this is explained in Section 2.6 by Klein et al. (2022). Using `conditional = FALSE` with probit-only models gives the same likelihood but different parameter estimates.
`theta`	an optional vector of starting values.
`fixed`	an optional named numeric vector of predefined parameter values or a logical (for `coef`) indicating to also return fixed parameters (only when `type = "all"`).
`scale`	a logical indicating if (internal) scaling shall be applied to the model coefficients.
`optim`	a list of optimisers as returned by `mltoptim`
`args`	a list of arguments for `lpmvnorm`.
`fit`	character vector describing how to fit the model. The default is joint likelihood estimation of all parameters, `pseudo` fixes the marginal parameters, `sequential` starts with a univariate model and sequentially adds models, keeping the parameters of previously added models fit. `ACS` implements Alternate Convex Search, starting with `pseudo` and, in a second step, fixing the marginal parameters. This is iterated for `ACSiter` iterations.
`ACSiter`	number of iterations for `fit = "ACS"`.

Details

The function implements multivariate conditional transformation models as described by Klein et al (2020). Below is a simple example for an unconditional bivariate distribution. See `demo("undernutrition", package = "tram")` for a conditional three-variate example.

Value

An object of class `Mmlt` with `coef` and `predict` methods.

References

Nadja Klein, Torsten Hothorn, Luisa Barbanti, Thomas Kneib (2022), Multivariate Conditional Transformation Models. *Scandinavian Journal of Statistics*, **49**, 116–142, doi:10.1111/sjos.12501.

Torsten Hothorn (2024), On Nonparanormal Likelihoods. doi:10.48550/arXiv.2408.17346.

Examples

```
data("cars")

### fit unconditional bivariate distribution of speed and distance to stop
## fit unconditional marginal transformation models
m_speed <- BoxCox(speed ~ 1, data = cars, support = ss <- c(4, 25),
                  add = c(-5, 5))
m_dist <- BoxCox(dist ~ 1, data = cars, support = sd <- c(0, 120),
                 add = c(-5, 5))

## fit multivariate unconditional transformation model
m_speed_dist <- Mmlt(m_speed, m_dist, formula = ~ 1, data = cars)

## log-likelihood
logLik(m_speed_dist)
sum(predict(m_speed_dist, newdata = cars, type = "density", log = TRUE))

## Wald test of independence of speed and dist (the "dist.speed.(Intercept)"
## coefficient)
summary(m_speed_dist)

## LR test comparing to independence model
LR <- 2 * (logLik(m_speed_dist) - logLik(m_speed) - logLik(m_dist))
pchisq(LR, df = 1, lower.tail = FALSE)

## constrain lambda to zero and fit independence model
## => log-likelihood is the sum of the marginal log-likelihoods
mI <- Mmlt(m_speed, m_dist, formula = ~1, data = cars,
           fixed = c("dist.speed.(Intercept)" = 0))
logLik(m_speed) + logLik(m_dist)
logLik(mI)

## linear correlation, ie Pearson correlation of speed and dist after
## transformation to bivariate normality
(r <- coef(m_speed_dist, type = "Corr"))

## Spearman's rho (rank correlation) of speed and dist on original scale
(rs <- coef(m_speed_dist, type = "Spearman"))

## evaluate joint and marginal densities (needs to be more user-friendly)
nd <- expand.grid(c(nd_s <- mkgrid(m_speed, 100), nd_d <- mkgrid(m_dist, 100)))
nd$d <- predict(m_speed_dist, newdata = nd, type = "density")

## compute marginal densities
nd_s <- as.data.frame(nd_s)
nd_s$d <- predict(m_speed_dist, newdata = nd_s, margins = 1L,
                  type = "density")
nd_d <- as.data.frame(nd_d)
nd_d$d <- predict(m_speed_dist, newdata = nd_d, margins = 2L,
                  type = "density")

## plot bivariate and marginal distribution
```

```
col1 <- rgb(.1, .1, .1, .9)
col2 <- rgb(.1, .1, .1, .5)
w <- c(.8, .2)
layout(matrix(c(2, 1, 4, 3), nrow = 2), width = w, height = rev(w))
par(mai = c(1, 1, 0, 0) * par("mai"))
sp <- unique(nd$speed)
di <- unique(nd$dist)
d <- matrix(nd$d, nrow = length(sp))
contour(sp, di, d, xlab = "Speed (in mph)", ylab = "Distance (in ft)", xlim = ss, ylim = sd,
        col = col1)
points(cars$speed, cars$dist, pch = 19, col = col2)
mai <- par("mai")
par(mai = c(0, 1, 0, 1) * mai)
plot(d ~ speed, data = nd_s, xlim = ss, type = "n", axes = FALSE,
     xlab = "", ylab = "")
polygon(nd_s$speed, nd_s$d, col = col2, border = FALSE)
par(mai = c(1, 0, 1, 0) * mai)
plot(dist ~ d, data = nd_d, ylim = sd, type = "n", axes = FALSE,
     xlab = "", ylab = "")
polygon(nd_d$d, nd_d$dist, col = col2, border = FALSE)

### NOTE: marginal densities are NOT normal, nor is the joint
### distribution. The non-normal shape comes from the data-driven
### transformation of both variables to joint normality in this model.
```

`mtram` *Transformation Models for Clustered Data*

Description

Marginally interpretable transformation models for clustered data.

Usage

```
mtram(object, formula, data,
      grd = SparseGrid::createSparseGrid(type = "KPU",
                 dimension = length(rt$cnms[[1]]), k = 10),
      tol = .Machine$double.eps, optim = mltoptim(auglag = list(maxtry = 5)),
      ...)
```

Arguments

`object`	A `tram` object.
`formula`	A formula specifying the random effects.
`data`	A data frame.
`grd`	A sparse grid used for numerical integration to get the likelihood.
`tol`	numerical tolerance.

`optim`	a list of optimisers as returned by `mltoptim`
`...`	Additional argument.

Details

A Gaussian copula with a correlation structure obtained from a random intercept or random intercept / random slope model (that is, clustered or longitudinal data can by modelled only) is used to capture the correlations whereas the marginal distributions are described by a transformation model. The methodology is described in Barbanti and Hothorn (2022) and examples are given in the `mtram` package vignette.

Only `coef()` and `logLik()` methods are available at the moment, see `vignette("mtram", package = "tram")` for worked examples.

Value

An object of class `tram` with `coef()` and `logLik()` methods.

References

Luisa Barbanti and Torsten Hothorn (2024). A Transformation Perspective on Marginal and Conditional Models, *Biostatistics*, **25**(2), 402–428, doi:10.1093/biostatistics/kxac048.

See Also

`vignette("mtram", package = "tram")`

Examples

```
if (require("lme4")) {
    ### linear mixed model
    sleep_lmer <- lmer(Reaction ~ Days + (Days | Subject),
                       data = sleepstudy, REML = FALSE)

    ### marginal transformation model
    sleep_LM <- Lm(Reaction ~ Days, data = sleepstudy)
    sleep_LMmer <- mtram(sleep_LM, ~ (Days | Subject), data = sleepstudy)

    ### the same
    logLik(sleep_lmer)
    logLik(sleep_LMmer)

    ### Lm / mtram estimate standardised effects
    sdinv <- 1 / summary(sleep_lmer)$sigma
    fixef(sleep_lmer) * c(-1, 1) * sdinv
    coef(sleep_LMmer)[c("(Intercept)", "Days")]
}
```

---

perm_test *Permutation Transformation Tests*

Description

P-values for a parameter in a linear transformation model and corresponding confidence intervals obtained from by the permutation principle

Usage

```
perm_test(object, ...)
## S3 method for class 'tram'
perm_test(object, parm = names(coef(object)),
    statistic = c("Score", "Likelihood", "Wald"),
    alternative = c("two.sided", "less", "greater"),
    nullvalue = 0, confint = TRUE, level = .95,
    Taylor = FALSE, block_permutation = TRUE, maxsteps = 25, ...)
```

Arguments

`object`	an object of class `tram`
`parm`	a vector of names of parameters to be tested. These parameters must be present in `object`.
`statistic`	a character string specifying the statistic to be permuted. The default `Score` is the classical permutation test for the esiduals of a model excluding the parameter `parm`. Only available for `nullvalue = 0`, confidence intervals are not available. Permuting the likelihood or the model coefficients under the nullvalue is highly expermimental as are the corresponding confidence intervals.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`.
`nullvalue`	a number specifying an optional parameter used to form the null hypothesis.
`confint`	a logical indicating whether a confidence interval should be computed. Score confidence intervals are computed by default. A 1st order Taylor approximation to the Score statistc is used with `Taylor = TRUE` (in case numerical inversion of the score statistic fails, Wald-type confidence intervals relying from this approximation are returned) . For the remaining likelihood and Wald statistics, confidence intervals are highly experimental (and probably not worth looking at).
`level`	the confidence level.
`block_permutation`	a logical indicating wheather stratifying variables shall be interpreted as blocks defining admissible permutations.
`Taylor`	a logical requesting the use of a 1st order Taylor approximation when inverting the score statistic.

`maxsteps`	number of function evaluations when inverting the score statistic for computing confidence intervals.
`...`	additional arguments to `independence_test`.

Details

Permutation test for one single parameters in the linear predictor of `object` is computed. This parameters must be present in `object`. This is somewhat experimental and not recommended for serious practical use (yet!).

Value

An object of class `htest` or a list thereof. See `Coxph` for an example.

Examples

```
## Tritiated Water Diffusion Across Human Chorioamnion
## Hollander and Wolfe (1999, p. 110, Tab. 4.1)
diffusion <- data.frame(
    pd = c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46,
           1.15, 0.88, 0.90, 0.74, 1.21),
    age = factor(rep(c("At term", "12-26 Weeks"), c(10, 5)))
)

### plot the two quantile functions
boxplot(pd ~ age, data = diffusion)

### the Wilcoxon rank sum test, with a confidence interval
### for a median shift
wilcox.test(pd ~ age, data = diffusion, conf.int = TRUE, exact = TRUE)

### a corresponding parametric transformation model with a log-odds ratio
### difference parameter, ie a difference on the log-odds scale
md <- Colr(pd ~ age, data = diffusion)

### assess model fit by plotting estimated distribution fcts
agef <- sort(unique(diffusion$age))
col <- c("black", "darkred")
plot(as.mlt(md), newdata = data.frame(age = agef),
     type = "distribution", col = col)
legend("bottomright", col = col, lty = 1, legend = levels(agef),
       bty = "n", pch = 19)
## compare with ECDFs: not too bad (but not good, either)
npfit <- with(diffusion, tapply(pd, age, ecdf))
lines(npfit[[1]], col = col[1])
lines(npfit[[2]], col = col[2])

### Wald confidence interval
confint(md)

### Likelihood confidence interval
confint(profile(md))
```

```
### Score confidence interval
confint(score_test(md))
confint(score_test(md, Taylor = TRUE))

### exact permutation score test
(pt <- perm_test(md, confint = TRUE, distribution = "exact"))
(pt <- perm_test(md, confint = TRUE, distribution = "exact",
                 Taylor = TRUE))

### compare with probabilistic indices obtained from asht::wmwTest
if (require("asht", warn.conflicts = FALSE)) {
    print(wt2 <- wmwTest(pd ~ I(relevel(age, "At term")),
                    data = diffusion, method = "exact.ce"))
    ### as log-odds ratios
    print(PI(prob = wt2$conf.int))
    print(PI(prob = wt2$estimate))
}
```

`Polr` *Ordered Categorical Regression*

Description

Some regression models for ordered categorical responses

Usage

```
Polr(formula, data, subset, weights, offset, cluster, na.action = na.omit,
     method = c("logistic", "probit", "loglog", "cloglog", "cauchit"), ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.

`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset.
`method`	a character describing the link function.
`...`	additional arguments to `tram`.

Details

Models for ordered categorical responses reusing the interface of `polr`. Allows for stratification, censoring and trunction.

The model is defined with a negative shift term, thus `exp(coef())` is the multiplicative change of the odds ratio (conditional odds for reference divided by conditional odds of treatment or for a one unit increase in a numeric variable). Large values of the linear predictor correspond to large values of the conditional expectation response (but this relationship is nonlinear).

Value

An object of class `Polr`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
data("wine", package = "ordinal")

library("MASS")
polr(rating ~ temp + contact, data = wine)

Polr(rating ~ temp + contact, data = wine)
```

`robust_score_test` *Doubly Robust Transformation Score Test*

Description

Doubly robust p-values and confidence intervals for parameters in (stratified) linear (shift-scale) transformation models obtained using the tram generalised covariance measure test.

Usage

```
robust_score_test(object, ...)

## S3 method for class 'tram'
robust_score_test(
  object,
  parm = names(coef(object)),
  alternative = c("two.sided", "less", "greater"),
  nullvalue = 0,
  confint = FALSE,
  level = 0.95,
  ranger_args = NULL,
  ...
)
```

Arguments

`object`	an object of class '`tram`'
`...`	additional arguments, currently ignored.
`parm`	a vector of names of parameters to be tested. These parameters must be present in `object`
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"`, or `"less"`
`nullvalue`	a number specifying an optional parameter used to form the null hypothesis H_0: parm = nullvalue and defaults to zero
`confint`	a logical indicating whether to (numerically) invert the test to obtain a robust score confidence interval
`level`	the confidence level
`ranger_args`	arguments passed to `ranger` for the regression of the column in the design matrix corresponding to `parm` against all others

Details

For a (stratified) linear shift (-scale) transformation he tram-GCM test tests the hypothesis H0: `parm` = `nullvalue` by re-fitting the model under the null hypothesis, computing the score residuals (see `residuals.tram`), and running an additional regression of the column in the design matrix corresponding to `parm` on the remaining columns, computing the corresponding residuals, and finally computing correlation-type test between the score and predictor residuals.

Value

An object of class '`htest`' or a list thereof.

Author(s)

Lucas Kook, Torsten Hothorn

References

Kook, L., Saengkyongam, S., Lundborg, A. R., Hothorn, T., & Peters, J. (2024). Model-based causal feature selection for general response types. Journal of the American Statistical Association, 1-12. doi:10.1080/01621459.2024.2395588

Examples

```
data("mtcars")
### Linear shift tram
m <- Lm(mpg ~ cyl + disp, data = mtcars)
robust_score_test(m, parm = "cyl")
### Linear shift-scale tram
m2 <- Lm(mpg ~ cyl | disp, data = mtcars)
robust_score_test(m2, parm = "cyl")
robust_score_test(m2, parm = "scl_disp")
### Stratified linear shift tram
m3 <- Lm(mpg | 0 + disp ~ cyl, data = mtcars)
robust_score_test(m3, parm = "cyl")
### Stratified linear shift-scale tram
m4 <- Lm(mpg | 0 + disp ~ cyl | cyl, data = mtcars)
robust_score_test(m4, parm = "cyl")
```

`score_test` *Transformation Score Tests and Confidence Intervals*

Description

P-values and confidence intervals for parameters in linear transformation models obtained from by the score test principle

Usage

```
score_test(object, ...)
## S3 method for class 'tram'
score_test(object, parm = names(coef(object)),
    alternative = c("two.sided", "less", "greater"), nullvalue = 0,
    confint = TRUE, level = .95, Taylor = FALSE, maxsteps = 25, ...)
```

Arguments

`object`	an object of class `tram`
`parm`	a vector of names of parameters to be tested. These parameters must be present in `object`.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`.
`nullvalue`	a number specifying an optional parameter used to form the null hypothesis.

`confint`	a logical indicating whether a confidence interval should be computed. Score confidence intervals are computed by default. A 1st order Taylor approximation to the Score statistc is used with `Taylor = TRUE` (in case numerical inversion of the score statistic fails, Wald confidence intervals relying from this approximation are returned).
`level`	the confidence level.
`Taylor`	a logical requesting the use of a 1st order Taylor approximation when inverting the score statistic.
`maxsteps`	number of function evaluations when inverting the score statistic for computing confidence intervals.
`...`	additional arguments, currently ignored.

Details

Score tests and confidence intervals for the parameters in the linear predictor of `object` are computed. These parameters must be present in `object`.

Value

An object of class `htest` or a list thereof. See `Coxph` for an example. A corresponding permutation test for parameters in a transformation models is available in `perm_test`.

`Survreg` *Parametric Survival Models*

Description

Weibull, log-normal, log-logistic and other parametric models (not exclusively) for survival analysis

Usage

```
Survreg(formula, data, subset, weights, offset, cluster, na.action = na.omit,
        dist = c("weibull", "logistic", "gaussian", "exponential", "rayleigh",
                 "loggaussian", "lognormal", "loglogistic"), scale = 0, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under `tram` and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.

`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset.
`dist`	character defining the conditional distribution of the (not necessarily positive) response, current choices include Weibull, logistic, normal, exponential, Rayleigh, log-normal (same as log-gaussian), or log-logistic.
`scale`	a fixed value for the scale parameter(s).
`...`	additional arguments to `tram`.

Details

Parametric survival models reusing the interface of `survreg`. The parameterisation is, however, a little different, see the package vignette.

The model is defined with a negative shift term. Large values of the linear predictor correspond to large values of the conditional expectation response (but this relationship is nonlinear). Parameters are log-hazard ratios comparing a reference with treatment (or a one unit increase in a numeric variable).

Value

An object of class `Survreg`, with corresponding `coef`, `vcov`, `logLik`, `estfun`, `summary`, `print`, `plot` and `predict` methods.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Examples

```
data("GBSG2", package = "TH.data")

library("survival")
survreg(Surv(time, cens) ~ horTh, data = GBSG2)

Survreg(Surv(time, cens) ~ horTh, data = GBSG2)
```

`tram` *Stratified Linear Transformation Models*

Description

Likelihood-inference for stratified linear transformation models, including linear shift-scale transformation models.

Usage

```
tram(formula, data, subset, weights, offset, cluster, na.action = na.omit,
     distribution = c("Normal", "Logistic", "MinExtrVal", "MaxExtrVal",
                      "Exponential", "Cauchy", "Laplace"),
     frailty = c("None", "Gamma", "InvGauss", "PositiveStable"),
     transformation = c("discrete", "linear", "logarithmic", "smooth"),
     LRtest = TRUE, prob = c(0.1, 0.9), support = NULL,
     bounds = NULL, add = c(0, 0), order = 6,
     negative = TRUE, remove_intercept = TRUE,
     scale = TRUE, scale_shift = FALSE, extrapolate = FALSE,
     log_first = FALSE, sparse_nlevels = Inf,
     model_only = FALSE, constraints = NULL, ...)
tram_data(formula, data, subset, weights, offset, cluster, na.action = na.omit)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model structure to be fitted. The details of model specification are given under Details and in the package vignette.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of case weights to be used in the fitting process. Should be `NULL` or a numeric vector. If present, the weighted log-likelihood is maximised.
`offset`	this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases.
`cluster`	optional factor with a cluster ID employed for computing clustered covariances.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set to `na.omit`.
`distribution`	character specifying how the transformation function is mapped into probabilities. Available choices include the cumulative distribution functions of the standard normal, the standard logistic and the standard minimum extreme value distribution.

`frailty`	character specifying the addition of a frailty term, that is, a random component added to the linear predictor of the model, with specific distribution (Gamma, inverse Gaussian, positive stable).
`transformation`	character specifying the complexity of the response-transformation. For discrete responses, one parameter is assigned to each level (except the last one), for continuous responses linear, log-linear and smooth (parameterised as a Bernstein polynomial) function are implemented.
`LRtest`	logical specifying if a likelihood-ratio test for the null of all coefficients in the linear predictor being zero shall be performed.
`prob`	two probabilities giving quantiles of the response defining the support of a smooth Bernstein polynomial (if `transformation = "smooth"`).
`support`	a vector of two elements; the support of a smooth Bernstein polynomial (if `transformation = "smooth"`).
`bounds`	an interval defining the bounds of a real sample space.
`add`	add these values to the support before generating a grid via `mkgrid`.
`order`	integer >= 1 defining the order of the Bernstein polynomial (if `transformation = "smooth"`).
`negative`	logical defining the sign of the linear predictor.
`remove_intercept`	logical defining if the intercept shall be removed from the linear shift predictor in favour of an (typically implicit) intercept in the baseline transformation. If `FALSE`, the linear shift predictor has an intercept (unless `-1` is added to the formula) but the baseline transformation is centered. For linear transformation models, this does not change the in-sample log-likelihood. For shift-scale transformation models, using `FALSE` ensures that centering of variables in the linear shift predictor does not affect the corresponding estimates and standard errors. Note that linear scale predictors are always fitted without intercept.
`scale`	logical defining if variables in the linear predictor shall be scaled. Scaling is internally used for model estimation, rescaled coefficients are reported in model output.
`scale_shift`	a logical choosing between two different model types in the presence of a `scaling` term, see `ctm`.
`extrapolate`	logical defining the behaviour of the Bernstein transformation function outside `support`. The default `FALSE` is to extrapolate linearily without requiring the second derivative of the transformation function to be zero at `support`. If `TRUE`, this additional constraint is respected.
`sparse_nlevels`	integer; use a sparse model matrix if the number of levels of an ordered factor is at least as large as `sparse_nlevels`.
`log_first`	logical; if `TRUE`, a Bernstein polynomial is defined on the log-scale.
`model_only`	logical, if `TRUE` the unfitted model is returned.
`constraints`	additional constraints on regression coefficients in the linear predictor of the form `lhs %*% coef(object) >= rhs`, where `lhs` and `rhs` can be specified as a character (as in `glht`) or by a matrix `lhs` (assuming `rhs = 0`), or as a list containing the two elements `lhs` and `rhs`.
`...`	additional arguments.

Details

The model formula is of the form `y | s ~ x | z` where `y` is an at least ordered response variable, `s` are the variables defining strata and `x` defines the linear predictor. Optionally, `z` defines a scaling term (see `ctm`). `y ~ x` defines a model without strata (but response-varying intercept function) and `y | s ~ 0` sets-up response-varying coefficients for all variables in `s`.

The two functions `tram` and `tram_data` are not intended to be called directly by users. Instead, functions `Coxph` (Cox proportional hazards models), `Survreg` (parametric survival models), `Polr` (models for ordered categorical responses), `Lm` (normal linear models), `BoxCox` (non-normal linear models) or `Colr` (continuous outcome logistic regression) allow direct access to the corresponding models.

The model class and the specific models implemented in **tram** are explained in the package vignette of package **tram**. The underlying theory of most likely transformations is presented in Hothorn et al. (2018), computational and modelling aspects in more complex situations are discussed by Hothorn (2018).

Value

An object of class `tram` inheriting from `mlt`.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Torsten Hothorn (2020), Most Likely Transformations: The mlt Package, *Journal of Statistical Software*, **92**(1), doi:10.18637/jss.v092.i01.

Sandra Siegfried, Lucas Kook, Torsten Hothorn (2023), Distribution-Free Location-Scale Regression, *The American Statistician*, doi:10.1080/00031305.2023.2203177.

Examples

```
data("BostonHousing2", package = "mlbench")

### unconstrained regression coefficients
### BoxCox calls tram internally
m1 <- BoxCox(cmedv ~ chas + crim + zn + indus + nox +
             rm + age + dis + rad + tax + ptratio + b + lstat,
             data = BostonHousing2)

### now with two constraints on regression coefficients
m2 <- BoxCox(cmedv ~ chas + crim + zn + indus + nox +
             rm + age + dis + rad + tax + ptratio + b + lstat,
             data = BostonHousing2,
             constraints = c("crim >= 0", "chas1 + rm >= 1.5"))
coef(m1)
coef(m2)

K <- matrix(0, nrow = 2, ncol = length(coef(m2)))
colnames(K) <- names(coef(m2))
K[1, "crim"] <- 1
```

```
K[2, c("chas1", "rm")] <- 1
m3 <- BoxCox(cmedv ~ chas + crim + zn + indus + nox +
             rm + age + dis + rad + tax + ptratio + b + lstat,
             data = BostonHousing2,
             constraints = list(K, c(0, 1.5)))
all.equal(coef(m2), coef(m3))
```

`tram-methods` *Methods for Stratified Linear Transformation Models*

Description

Methods for objects inheriting from class tram

Usage

```
## S3 method for class 'tram'
as.mlt(object)
## S3 method for class 'tram'
model.frame(formula, ...)
## S3 method for class 'tram'
model.matrix(object, data = object$data, with_baseline = FALSE,
       what = c("shifting", "interacting"), ...)
## S3 method for class 'stram'
model.matrix(object, data = object$data, with_baseline = FALSE,
       what = c("shifting", "scaling", "interacting"), ...)
## S3 method for class 'tram'
coef(object, with_baseline = FALSE, ...)
## S3 method for class 'Lm'
coef(object, as.lm = FALSE, ...)
## S3 method for class 'Survreg'
coef(object, as.survreg = FALSE, ...)
## S3 method for class 'tram'
vcov(object, with_baseline = FALSE, complete = FALSE, ...)
## S3 method for class 'tram'
logLik(object, parm = coef(as.mlt(object), fixed = FALSE), ...)
## S3 method for class 'tram'
estfun(x, parm = coef(as.mlt(x), fixed = FALSE), ...)
## S3 method for class 'tram'
predict(object, newdata = model.frame(object),
        type = c("lp", "trafo", "distribution", "logdistribution",
             "survivor", "logsurvivor", "density", "logdensity",
             "hazard", "loghazard", "cumhazard", "logcumhazard",
             "odds", "logodds", "quantile"), ...)
## S3 method for class 'stram'
predict(object, newdata = model.frame(object),
```

```
        type = c("lp", "trafo", "distribution", "logdistribution",
             "survivor", "logsurvivor", "density", "logdensity",
             "hazard", "loghazard", "cumhazard", "logcumhazard",
             "odds", "logodds", "quantile"),
        what = c("shifting", "scaling"), ...)
## S3 method for class 'tram'
plot(x, newdata = model.frame(x),
     which = c("QQ-PIT", "baseline only", "distribution"),
     confidence = c("none", "interval", "band"), level = 0.95,
     K = 50, cheat = K, col = "black", fill = "lightgrey", lwd = 1, ...)
## S3 method for class 'tram'
residuals(object, ...)
## S3 method for class 'tram'
PI(object, newdata = model.frame(object), reference = 0,
                  one2one = FALSE, ...)
## Default S3 method:
PI(object, prob, link = "logistic", ...)
## S3 method for class 'tram'
OVL(object, newdata = model.frame(object), reference = 0,
                  one2one = FALSE, ...)
## Default S3 method:
OVL(object, link = "logistic", ...)
## S3 method for class 'tram'
TV(object, newdata = model.frame(object), reference = 0,
                  one2one = FALSE, ...)
## Default S3 method:
TV(object, link = "logistic", ...)
## S3 method for class 'tram'
L1(object, newdata = model.frame(object), reference = 0,
                  one2one = FALSE, ...)
## Default S3 method:
L1(object, link = "logistic", ...)
## S3 method for class 'tram'
ROC(object, newdata = model.frame(object), reference = 0,
                   prob = 1:99 / 100, one2one = FALSE, ...)
## Default S3 method:
ROC(object, prob = 1:99 / 100, link = "logistic", ...)
## S3 method for class 'ROCtram'
plot(x, lty = 1:ncol(x), col = "black",
     fill = "lightgrey", lwd = 1, ...)
```

Arguments

`object`, `formula`, `x`	a fitted stratified linear transformation model inheriting from class `tram`. `PI` also takes a numeric vector in the default method.
`data`	an optional data frame.
`with_baseline`	logical, if `TRUE` all model parameters are returned, otherwise parameters describ-

	ing the baseline transformation are ignored.
`as.lm`	logical, return parameters in the `lm` parameterisation if `TRUE`.
`as.survreg`	logical, return parameters in the `survreg` parameterisation in `TRUE`.
`parm`	model parameters, including baseline parameters.
`complete`	currently ignored
`newdata`	an optional data frame of new observations.
`reference`	an optional data frame of reference observations, or a numeric vector of reference values.
`type`	type of prediction, current options include linear predictors (`"lp"`, of `x` variables in the formula `y | s ~ x`), transformation functions (`"trafo"`) or distribution functions on the scale of the cdf (`"distribution"`), survivor function, density function, log-density function, hazard function, log-hazard function, cumulative hazard function or quantile function.
`which`	type of plot, either a QQ plot of the probability-integral transformed observations (`"QQ-PIT"`), of the baseline transformation of the whole distribution.
`what`	type of model matrix / linear predictor: `shifting` returns model model matrix / linear predictor for shift term, `scaling` for the scale term.
`confidence`	type of uncertainty assessment.
`level`	confidence level.
`K`	number of grid points in the response, see `plot.ctm`.
`cheat`	reduced number of grid points for the computation of confidence bands, see `confband`.
`col`	line color.
`fill`	fill color.
`lwd`	line width.
`lty`	line type.
`prob`	a numeric vector of probabilities..
`link`	a character identifying a link function.
`one2one`	logical, compute the ROC curve (and derived measures) comparing each row in `newdata` with each row in `reference` (`FALSE`, the default), or compare observations rowwise (`TRUE`).
`...`	additional arguments to the underlying methods for class `mlt`, see `mlt-methods`.

Details

`coef` can be used to get (and set) model parameters, `logLik` evaluates the log-likelihood (also for parameters other than the maximum likelihood estimate); `vcov` returns the estimated variance-covariance matrix (possibly taking `cluster` into account) and and `estfun` gives the score contribution by each observation. `predict` and `plot` can be used to inspect the model on different scales.

`PI` computes the probabilistic index (or concordance probability or AUC) for all observations in `newdata`, relative to `reference`, ie the probability

$$P(Y_1 \le Y_0 \mid x_0, x_1)$$

of observing a smaller value of a randomly sampled observation conditional on x_1 compared to a randomly sampled reference observation, which is conditional on x_0. This is equivalent to the area under the receiver operating curve (ROC). The probability only applies within strata, response-varying coefficients are not allowed.

Under the same setup, `OVL` gives the overlap coefficient, which is one minus the total variation and one minus half the L_1 distance between the two conditional densities. The overlap coefficient is identical to the Youden index and the Smirnov statistic.

`PI` and friends also accept an argument `conf.level` which triggers computation of simultaneous Wald confidence intervals for these measures. Arguments in ... are forwarded to `glht`.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

See Also

`mlt-methods`, `plot.ctm`

Examples

```
data("BostonHousing2", package = "mlbench")

### fit non-normal Box-Cox type linear model with two
### baseline functions (for houses near and off Charles River)
BC_BH_2 <- BoxCox(cmedv | 0 + chas ~ crim + zn + indus + nox +
                  rm + age + dis + rad + tax + ptratio + b + lstat,
                  data = BostonHousing2)
logLik(BC_BH_2)

### classical likelihood inference
summary(BC_BH_2)

### coefficients of the linear predictor
coef(BC_BH_2)

### plot linear predictor (mean of _transformed_ response)
### vs. observed values
plot(predict(BC_BH_2, type = "lp"), BostonHousing2$cmedv)

### all coefficients
coef(BC_BH_2, with_baseline = TRUE)

### compute predicted median along with 10% and 90% quantile for the first
### observations
predict(BC_BH_2, newdata = BostonHousing2[1:3,], type = "quantile",
        prob = c(.1, .5, .9))

### plot the predicted density for these observations
plot(BC_BH_2, newdata = BostonHousing2[1:3, -1],
     which = "distribution", type = "density", K = 1000)
```

```
### evaluate the two baseline transformations, with confidence intervals
nd <- model.frame(BC_BH_2)[1:2, -1]
nd$chas <- factor(c("0", "1"))
library("colorspace")
col <- diverge_hcl(2, h = c(246, 40), c = 96, l = c(65, 90))
fill <- diverge_hcl(2, h = c(246, 40), c = 96, l = c(65, 90), alpha = .3)
plot(BC_BH_2, which = "baseline only", newdata = nd, col = col,
     confidence = "interval", fill = fill, lwd = 2,
     xlab = "Median Value", ylab = expression(h[Y]))
legend("bottomright", lty = 1, col = col,
        title = "Near Charles River", legend = c("no", "yes"), bty = "n")

### cars data; with quantile functions
plot(dist ~ speed, data = cars)
m <- Colr(dist ~ speed, data = cars)
q <- predict(as.mlt(m), newdata = data.frame(speed = s <- 6:25),
             type = "quantile", prob = c(1, 5, 9) / 10)
lines(s, q[1,])
lines(s, q[2,])
lines(s, q[3,])

nd <- data.frame(speed = s <- as.double(1:5 * 5))

# Prob(dist at speed s > dist at speed 0)
# speed 0 is reference, not a good choice here
PI(m, newdata = nd)

# Prob(dist at speed s > dist at speed 15)
lp15 <- c(predict(m, newdata = data.frame(speed = 15)))
PI(m, newdata = nd, reference = lp15)
PI(m, newdata = nd, reference = nd[3,,drop = FALSE])

# Prob(dist at speed s' > dist at speed s)
PI(m, newdata = nd, reference = nd)
# essentially:
lp <- predict(m, newdata = nd)
PI(object = dist(lp))
# same, with simultaneous confidence intervals
PI(m, newdata = nd, reference = nd, conf.level = .95)

# plot ROC curves + confidence bands
# compare speed 20 and 25 to speed 15
plot(ROC(m, newdata = nd[4:5,,drop = FALSE],
         reference = nd[3,,drop = FALSE],
         conf.level = 0.95))

# Overlap of conditional densities at speed s' and s
OVL(m, newdata = nd, reference = nd)

### ROC analysis (takes too long for CRAN Windows)
if (require("mlbench") && .Platform$OS.type != "windows") {
```

```
    layout(matrix(1:4, nrow = 2))
    data("PimaIndiansDiabetes2", package = "mlbench")
    dia <- sort(unique(PimaIndiansDiabetes2$diabetes))
    nd <- data.frame(diabetes = dia,
                     age = 29, mass = 32) ### median values

    ### unconditional ROC analysis: glucose tolerance test
    m0 <- Colr(glucose ~ diabetes, data = PimaIndiansDiabetes2)
    # ROC curve + confidence band
    plot(ROC(m0, newdata = nd[2,,drop = FALSE], conf.level = .95))
    # Wald interval for AUC
    PI(m0, newdata = nd[2,,drop = FALSE], conf.level = .95)
    # score interval for AUC
    PI(-c(coef(m0), score_test(m0)$conf.int[2:1]))

    ### adjusted ROC analysis for age and mass
    m1 <- Colr(glucose ~ diabetes + age + mass, data = PimaIndiansDiabetes2)
    # ROC curve + confidence band (this is the same for all ages /
    # masses)
    plot(ROC(m1, newdata = nd[2,,drop = FALSE],
                 reference = nd[1,,drop = FALSE],
             conf.level = .95))
    # Wald interval for adjusted AUC
    PI(m1, newdata = nd[2,,drop = FALSE], reference = nd[1,,drop = FALSE],
       conf.level = .95)
    # Score interval for adjusted AUC
    PI(-c(coef(m1)[1], score_test(m1, names(coef(m1))[1])$conf.int[2:1]))

    ### conditional ROC analysis: AUC regression ~ age + mass
    m2 <- Colr(glucose ~ diabetes * (age + mass), data = PimaIndiansDiabetes2)
    # ROC curve for a person with age = 29 and mass = 32
    plot(ROC(m2, newdata = nd[2,,drop = FALSE],
                 reference = nd[1,,drop = FALSE],
             conf.level = .95))
    # AUC for persons ages 21:81, all with mass = 32
    nd1 <- data.frame(diabetes = nd[1,"diabetes"], age = 21:81, mass = 32)
    nd2 <- data.frame(diabetes = nd[2,"diabetes"], age = 21:81, mass = 32)
    auc <- PI(m2, newdata = nd2, reference = nd1, one2one = TRUE,
              conf.level = 0.95)
    plot(nd1$age, auc[, "Estimate"], xlab = "Age (in years)", ylab =
         "AUC", ylim = c(0, 1), type = "l")
    lines(nd1$age, auc[, "lwr"], lty = 3)
    lines(nd1$age, auc[, "upr"], lty = 3)
}
```

Index