Package: transitreg (via r-universe)

May 27, 2026

Title Flexible Transition Models for Probabilistic Learning

Version 0.2-1

Date 2026-04-15

Description Provides infrastructure for estimating flexible transition models, supporting both count and continuous responses. For continuous responses, a slicing trick is applied to estimate fully probabilistic models.

License GPL-2 | GPL-3

Depends R (>= 4.1.0), mgcv, topmodels, distributions3

Imports Formula

Suggests colorspace, nnet, quarto, glmnet, keras3, tensorflow

VignetteBuilder quarto

LazyData yes

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository https://retostauffer.r-universe.dev

Date/Publication 2026-05-27 09:16:50 UTC

RemoteUrl https://github.com/retostauffer/transitreg

RemoteRef HEAD

RemoteSha 9c22f038950802a422fde7bdce353bcbc32eb957

Contents

`check_and_prepare_newdata` *Check and Prepare Newdata*

Description

If `newdata` are provided when calling the predict method, this function is used to (i) reduce `newdata` to the covariates originally used to estimate the model, and if factor checks all levels are known (i.e., all levels have been seen in the training data). TODO(R): This ensures we do not forward missing values to C which (currently) would result in a segfault as NA handling is not properly implemented – though it already checks for it and throws errors).

Usage

```
check_and_prepare_newdata(newdata, object)
```

Arguments

`newdata`	data.frame with the `newdata` for the prediction.
`mf`	the model frame of the model (i.e., training data).

Value

Returns a modified version of `newdata` (a subset) or throws errors if covariates are missing, or a class/level mismatch is found.

Author(s)

Reto

convert_tp *Converting between Densities, Probabilities, and Transition Probabilities*

Description

A transition model consists of a series of transition probabilities (TP) which define the probability that an observation falls into the next higher bin. Based on these transition probabilities, the cumulative distribution function (CDF) as well as the density (PDF) can be calculated. This utility function allows to convert from CDF to TPs, and from TPs back to CDF/PDF.

Usage

```
convert_tp(x, from, to, width = NULL, drop = TRUE)
```

Arguments

x	Numeric vector with density, probabilities, or transition probabilities. All non-missing >=0.
from	Character, current type of data in x. Either "tp", "cdf", or "pdf".
to	Character, type into which x should be converted (see from). Note that not all combinations of conversions are allowed. If not possible, an error will be thrown.
width	NULL or numeric (either length 1 or same length as x. Width of the individual bins represented in x.
drop	If TRUE (default) the result is simplified if possible.

Details

If from = "tp" (transition probabilities) all values in x must be in [0, 1] and monotonically decreasing. Transition probabilities can be converted into both, to = "cdf" as well as to = "pdf" (or both at the same time, setting to = c("cdf", "pdf")‘.

If from = "cdf" (distribution) all elements in x must be in [0,1]. Can be converted to = "tp".

Similarly, from = "pdf" can only be converted to = "tp". This requires all elements in x to be in [0, Inf].

Value

If to is a single character and drop = TRUE, the return is a numeric vector of the same length as the input argument x. Else the return is a data.frame with the same number of rows as length(x).

Author(s)

Reto

Examples

```
## For testing
## Draw PDF and CDF from Poisson distribution
p <- ppois(0:10, lambda = 4)
d <- dpois(0:10, lambda = 4)

## Convert Poisson CDF to transition probabilities
tp <- convert_tp(p, from = "cdf", to = "tp")
round(tp, 2)

pd <- convert_tp(tp, from = "tp", to = c("pdf", "cdf"))

## Convert transition probabilities back to CDF
p2 <- convert_tp(tp, from = "tp", to = "cdf")
all.equal(p, p2) # Must be equal

## Convert transition probabilities to PDF
d2 <- convert_tp(tp, from = "tp", to = "pdf")
all.equal(d, d2) # Must be equal

## Or directly from tp to both, CDF and PDF
d2 <- convert_tp(tp, from = "tp", to = c("cdf", "pdf"))
all.equal(p, d2$cdf)
all.equal(d, d2$pdf)

## Quick visual representation
barplot(tp, col = "steelblue", main = "Transition Probabilities")
col <- c("gray80", "tomato")
barplot(rbind(p, p2), beside = TRUE, main = "CDF Comparison", col = col)
barplot(rbind(d, d2$pdf), beside = TRUE, main = "PDF Comparison", col = col)
```

`get_elementwise_colnames`

Create Column Names for Return Matrix

Description

Used in the S3 methods pdf, cdf, and quantile. If `elementwise = FALSE` the return is a matrix; this helper function creates the names based on the thresholds/probabilities used.

Usage

```
get_elementwise_colnames(
  x,
  prefix = NULL,
  digits = pmax(3L, getOption("digits") - 3L)
)
```

Arguments

x	numeric, thresholds (pdf, cdf) or probabilities (quantile).
prefix	If NULL (quantiles) the result is in percent, else this prefix is used for each 'x'.
digits	Integer, number of significant digits for names.

Details

If prefix = NULL it is expected that x contains probabilities in the range of [0,1]. The returned vector of names will therefore contain "10%", "20%", "99%" etc.

When a prefix is set, x is interpreted as numeric value and the returned names will be a combination of the prefix and the numeric value of x. This will result in e.g, p_-12.4, p_0, p_11.302, ...

Value

A character vector with 'column names'.

Author(s)

Reto

---

make_breaks *Create Breaks for (Pseudo-)bins*

Description

Calculates the breaks (point intersection between breaks) which span the range of y if the user did not specify a vector of breaks, but only a single numeric (integer).

Usage

```
make_breaks(y, breaks, censored)
```

Arguments

y	a numeric vector with response data.
breaks	a single numeric (integer) >= 1, number of breaks to be created.
censored	character, one of "uncensored", "left", "right".

Value

Returns a numeric vector with the breaks.

num2bin *Convert Numerics to Pseudo-Bins*

Description

Convert response on original scale to (pseudo-)bin index.

Usage

```
num2bin(
  x,
  breaks = NULL,
  censored = c("uncensored", "left", "right", "both"),
  verbose = FALSE
)
```

Arguments

x	numeric vector.
breaks	numeric vector with point intersections to create the bins.
censored	character, defines if the (pseudo-)bins are censored or not (see 'Details' section).
verbose	logical, if TRUE some messages are printed if needed (defaults to FALSE).

Details

Converts the numeric values in x into (pseudo-)bins by cutting the data into length(breaks) - 1 segments. If censored = "uncensored" (no censoring), data outside range(break) will be set to -1L if x < min(breaks) or length(breaks) - 1L if x > max(breaks).

If censored = "left" all values x <= min(breaks) are assigned to as bin 0L, followed by bin 1L for (breaks[1], breaks[2]), 2L for [breaks[2], breaks[3]) etc. Thus, the highest bin containing data is length(breaks) as well.

If censored = "right" all values x >= max(breaks) are set to the highest bin length(breaks), those < max(breaks) are still set NA_integer_.

When censored = "both" the rules above are combined. Due to the necessity that we need tone dedicated 'censored bin index' on the left (0L), the right most upper bin is shifted by +1 compared to when we only use censored = "right".

Less important to know for an end-user as this is mainly handelled internally.

Examples

```
x      <- 0:10
breaks <- c(2, 6, 8)
transitreg:::num2bin(x, breaks = breaks)
transitreg:::num2bin(x, breaks = breaks, censored = "left")
transitreg:::num2bin(x, breaks = breaks, censored = "right")
transitreg:::num2bin(x, breaks = breaks, censored = "both")
```

```
transitreg:::num2bin(x, breaks = breaks, censored = "both")
```

`plot.transitreg` *Plot Method for Transition Model Fits*

Description

Provides diagnostic and visualization plots for transition models fitted using the `transitreg()` function. The method supports plotting effects and quantile residual diagnostic plots.

Usage

```
## S3 method for class 'transitreg'
plot(x, which = "effects", ask = NULL, ...)
```

Arguments

`x`	An object of class `"transitreg"` resulting from a call to `transitreg()`
`which`	A character or integer, specifying the type of plot(s) to generate (See 'Details').
`ask`	Either `NULL`, `TRUE` or `FALSE`. If `NULL` it will evaluate to `TRUE` if more than one plot is requested via `which`.
`...`	Additional arguments passed to the underlying plotting functions.

Details

The function allows to control what to plot via the argument `which`. Options include:

- `"effects"` Effects of the predictors on the response. Requires that the model is estimated by `mgcv::gam()` or `mgcv::bam()`.
- `"rootogram"`: Rootogram for assessing goodness of fit.
- `"qqrplot"`: Q-Q plot for quantile residuals.
- `"wormplot"`: Worm plot for quantile residuals.
- `"pithist"`: Probability integral transform (PIT) histogram.

Multiple options can be specified as a character vector or numeric indices.

The `plot.transitreg()` method provides flexible visualization options for evaluating transition model fits. Users can choose to:

- Visualize the effects of predictors on the response variable (if the model is a GAM, see `mgcv::gam()`).
- Evaluate quantile residuals through histograms, Q-Q plots, or worm plots employing the `topmodels` framework for model evaluation.

The `which` argument controls the type of plots generated. By default, the `"effects"` plot is shown if the model supports it. Residual-based plots (`"rootogram"`, `"qqrplot"`, `"wormplot"`, `"pithist"`) provide insights into the goodness of fit of the model and model calibration.

Value

Returns NULL invisibly. Generates plots as a side effect.

See Also

transitreg(), predict.transitreg().

Examples

```
## Example: Fit a transition model and generate plots.
set.seed(123)
n <- 500
x <- runif(n, -3, 3)
y <- rpois(n, exp(2 + sin(x)))
b <- transitreg(y ~ s(theta) + s(x))

## Plot effects.
plot(b, which = "effects")

## Plot residuals.
plot(b, which = c("rootogram", "pithist", "qqrplot", "wormplot"))

## Custom plot layout.
par(mfrow = c(2, 2))
plot(b, which = 2:5, ask = FALSE)
```

---

predict.transitreg *Predict Method for Transition Model Fits*

Description

Provides predictions for transition models fitted using the transitreg() function. Predictions can be generated for the probability density function (PDF), cumulative distribution function (CDF), maximum probability, or specific quantiles of the response distribution.

Usage

```
## S3 method for class 'transitreg'
predict(
  object,
  newdata = NULL,
  y = NULL,
  prob = NULL,
  type = c("pdf", "cdf", "quantile", "mode", "mean", "tp", "survival"),
  ncores = NULL,
  elementwise = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

object	An object of class transitreg resulting from a call to transitreg().
newdata	Optional. A data frame containing new predictor values. If not provided, predictions are made for the data used in fitting the model.
y	Optional. A vector of response values for which the PDF or CDF should be computed. Required if type is "pdf" or "cdf".
prob	Optional. A numeric value specifying the quantile to compute when type = "quantile". Default is 0.5 (median). If provided, argument type is set to type = "quantile".
type	Character. Specifies the type of prediction to return (see Section 'Details').
ncores	NULL (default) or single numeric. See section 'OpenMP' of the transitreg() man page for more details.
elementwise	Logical. Should each distribution in object be evaluated at all elements of prob/y? (elementwise = FALSE, yielding a matrix)? Or, if x and probs have the same length, should the evaluation be done element by element (elementwise = TRUE yielding a vector)? The default of NULL means that elementwise = TRUE is used if the lengths match and otherwise elementwise = FALSE is used.
verbose	Logical, if TRUE few messages will be printed.
...	Additional arguments passed to the prediction function.

Details

The predict.transitreg method computes predictions based on the transition model fit. Predictions can be made for the original training data or for new data provided via newdata. The method also supports scaling of covariates if scaling was applied during model fitting (argument scale.x in function transitreg()).

The argument type controls the return, the following types are allowed:

- "pdf": The predicted probability density function (PDF).
- "cdf": The cumulative distribution function (CDF).
- "survival": Survival function (1 - CDF).
- "mode": The expected value of the response (maximum probability).
- "mean": Expectation (weighted mean).
- "quantile": The quantile of the response specified by prob.

For "pdf" and "cdf", the response values (y) must be provided unless the model was fit with those values already included. For "quantile", a specific quantile(s) are computed based on prob.

Value

Returns predictions of the specified type:

- For "pdf" and "cdf", a vector or matrix of probabilities evaluated at y.
- For "mode", the expected value of the response.
- For "quantile", the quantile of the response distribution at the specified prob.

Author(s)

Niki

See Also

transitreg(), transitreg_tmf().

Examples

```
## Example: Predicting PDF and CDF.
set.seed(123)
n <- 500
x <- runif(n, -3, 3)
y <- rpois(n, exp(2 + sin(x)))
b <- transitreg(y ~ s(theta) + s(x))

## Predict PDF and CDF.
p <- list()
p$pdf <- predict(b, type = "pdf", y = 3)
p$cdf <- predict(b, type = "cdf", y = 3)

## Predict mode (expected value at highest probability)
p$mode <- predict(b, type = "mode")

## Predict quantiles.
p$qu95 <- predict(b, prob = 0.95)

print(head(as.data.frame(p)))

## Visualize predictions.
nd <- data.frame(x = seq(-3, 3, length = 100))

## Predict quantiles.
qu <- c(0.01, 0.05, 0.1, 0.2, 0.5, 0.7, 0.9, 0.95, 0.99)
p <- lapply(qu, function(prob) {
  predict(b, newdata = nd, prob = prob)
})

## Plot data and fitted quantiles.
plot(y ~ x, pch = 16, col = rgb(0.1, 0.1, 0.1, alpha = 0.4))
matplot(nd$x, do.call("cbind", p),
  type = "l", col = 4, lwd = 2, lty = 1,
  add = TRUE)
```

Shannon *Weather Data Shannon Airport (Ireland)*

Description

This data set contains daily meteorological observations from Shannon Airport, located on the west coast of Ireland, covering the period from September 1, 1945, to January 31, 2025 (29,008 days, approximately 79.5 years). The data, provided by the Irish Meteorological Service (Met Éireann), includes daily minimum and maximum temperatures, precipitation amounts, and sunshine duration.

Usage

```
Shannon
```

Format

A data frame containing historical meteorological observations from Shannon Airport (Clare County, Ireland) spanning over 79 years, without gaps or missing values. Observations are valid from 0000 UTC to 0000 UTC.

The data frame contains the following variables:

- `date`: Date of observation
- `maxtp`: Daily maximum temperature [°C]
- `mintp`: Daily minimum temperature [°C]
- `rain`: Daily precipitation sum [mm]
- `sun`: Sunshine duration [hours]

For more details about the observations and recording period, visit `https://www.met.ie/climate/what-we-measure`.

Details

Station information

- Country/county: Ireland, Clare
- Station: Shannon Airport (ID 518)
- Altitude: 15 m a.m.s.l.
- Location: 52.69028°N, 8.91806°W

For more details, see the weather observations website.

Author(s)

Reto

Source

Met Éireann, the Irish Meteorological Service, licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) license via. Available at `https://www.met.ie/climate/available-data/historical-data`.

Examples

```
data(Shannon)
summary(Shannon)
```

`survival` *Creating Survival Object*

Description

Creates a survival object used as response variable in a transitreg model formula.

Usage

```
survival(time, event)

## S3 method for class 'survival'
as.character(x, ...)

## S3 method for class 'survival'
print(x, quote = FALSE, ...)

## S3 method for class 'survival'
format(x, ...)

## S3 method for class 'survival'
x[i, ...]
```

Arguments

`time`	the time to event, the varaible the binning is taking place on.
`event`	must evaluate to logical, whether or not an event happened (`FALSE` equals survived, `TRUE` equals dead). If `event` is a single logical it is recycled for all observations, else the length of the two vectors `time` and `event` must match exactely.

Value

An object of class `survival` which can be used as response in transitreg models.

Author(s)

Reto

`Transition` *Creates a Transition Distribution*

Description

A 'Transition' distrubiton consists of a series of `K` transition probabilities (TP) for `K` 'bins' (counts or pseudo-counts).

Usage

```
Transition(x, breaks, censored = c("uncensored", "left", "right", "both"))

dtransit(x, d, log = FALSE, ncores = NULL)

ptransit(x, d, lower.tail = TRUE, log.p = FALSE, ncores = NULL)

qtransit(p, d, lower.tail = TRUE, log.p = FALSE, ncores = NULL)

rtransit(n, d, ncores = NULL)

## S3 method for class 'Transition'
c(...)

## S3 method for class 'Transition'
as.matrix(x, expand = FALSE, ...)

## S3 method for class 'Transition'
pdf(d, x, drop = TRUE, elementwise = NULL, ncores = NULL, ...)

## S3 method for class 'Transition'
log_pdf(d, x, drop = TRUE, elementwise = NULL, ncores = NULL, ...)

## S3 method for class 'Transition'
cdf(d, x, drop = TRUE, elementwise = NULL, ncores = NULL, ...)

## S3 method for class 'Transition'
quantile(
  x,
  probs,
  drop = TRUE,
  elementwise = NULL,
  ncores = NULL,
  approx = FALSE,
  ...
)

## S3 method for class 'Transition'
```

```
median(x, na.rm = NULL, ncores = NULL, ...)

## S3 method for class 'Transition'
mean(x, ncores = NULL, ...)

## S3 method for class 'Transition'
random(x, n = 1L, drop = TRUE, ...)

## S3 method for class 'Transition'
is_discrete(d, ...)

## S3 method for class 'Transition'
is_continuous(d, ...)

## S3 method for class 'Transition'
support(d, drop = NULL, ...)

## S3 method for class 'Transition'
plot(x, cdf = FALSE, tp = FALSE, all = FALSE, n = 8L, plot = TRUE, ...)
```

Arguments

x	Numeric, vector of quantiles.
breaks	numeric vector of points of intersection of the breaks. The length the vector must be of length(x) + 1 (if x is a vector) or ncol(x) + 1 if x is a matrix. Must be monotonically increasing. If breaks is integer, all must be >=0 (used to identify count data Transition distributions).
censored	character, or one of "uncensored" (default), "left", "right", or "both".
d	An object of class Transition.
log, log.p	Logical, if TRUE, probabilities p are given as log(p).
ncores	Number of cores to be used (see transitreg() for details).
lower.tail	Logical, if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].
p	Numeric, vector of probabilities.
n	Integer, maximum number of distributions to be plotted, defaults to 8L.
...	unused.
expand	logical, if FALSE (default) the wide format is returned, else the extended (long) form.
drop	Logical. Should the result be simplified to a vector if possible?
elementwise	Logical. Should each distribution in x be evaluated at all elements of probs (elementwise = FALSE, yielding a matrix)? Or, if x and probs have the same length, should the evaluation be done element by element (elementwise = TRUE yielding a vector)? The default of NULL means that elementwise = TRUE is used if the lengths match and otherwise elementwise = FALSE is used.
probs	numeric vector of probabilities with values in [0,1]. (Values up to '2e-14' outside that range are accepted and moved to the nearby endpoint.) TODO(R): SURE?

approx	logical, if FALSE (default) quantiles of the discrete distribution is returned. If TRUE the quantile is approximated, using linear interpolation. TODO(R): For development purposes only, decide if we want to keep this (and if so, check if correct).
na.rm	Unused.
cdf	logical. If cdf = TRUE then the cumulative distribution function (c.d.f.) is plotted. Otherwise, the probability density function (p.d.f.), for a continuous variable, or the probability mass function (p.m.f.), for a discrete variable, is plotted.
tp	logical. If tp = TRUE then the transition probabilities are plotted in addition to the cummulative distribution function or the probability mass function (see cdf).
all	logical. If TRUE all distributions in x will be drawn. Else only the first n ones (default is 8).
plot	logical. If FALSE not plot is produced.

Details

TODO(R): Missing

Value

Returns an object of class c("Transition", "distribution"). TODO(R): Missing.

Numeric matrix. If expand = FALSE the return is of dimension c(length(x), <ncol>).

If expand = TRUE the returned matrix is of dimension c(length(x) * <ncol>, 3L) where the four columns contain index (1, ..., length(x)) where each index corresponds to the row-index of the original input x (distribution identifier), theta which is the 'bin' the transition probability belongs to, followed by the transition probabilities tp itself. This expanded version is used when calling the .C functions.

Author(s)

Reto

Examples

```
# TODO(R) Write example
```

---

transitreg *Transition Models Estimation*

Description

Fits transition models to count or continuous response data. The method leverages transition probabilities to construct flexible probabilistic models without assuming a fixed distribution for the response variable. Estimation relies on **mgcv** infrastructure for GLM-type binary modeling.

Usage

```
transitreg(
  formula,
  data,
  subset,
  na.action,
  engine = "bam",
  breaks = NULL,
  censored = c("uncensored", "left", "right", "both"),
  model = TRUE,
  ncores = NULL,
  verbose = FALSE,
  ...
)

## S3 method for class 'transitreg'
prodist(object, newdata = NULL, ...)

## S3 method for class 'transitreg'
newresponse(object, newdata = NULL, ...)

## S3 method for class 'transitreg'
logLik(object, newdata = NULL, ...)
```

Arguments

formula	Object of class transitreg.
data	A data frame containing the data for modeling.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
na.action	A function that determines how NA values in the data should be handled.
engine	Character string specifying the estimation engine. Options include "bam", "gam", "nnet", or "glmnet" (experimental). Default is "bam".
breaks	Controls the splitting of continuous responses into intervals to mimic a count response. If a single number is provided, equidistant intervals are used. If a numeric vector is provided, it is used directly as the breakpoints. This argument is critical for continuous responses.
censored	character, or one of "uncensored" (default), "left", "right", or "both". Specifies if the distribution is uncensored or censored on one or both ends.
model	Logical value indicating whether the model frame should be included in the return object.
ncores	NULL (default) or single numeric. See 'OpenMP' for more information.
verbose	Logical value indicating whether progress and diagnostic messages should be printed. Default is FALSE.
...	Additional arguments to be passed to the estimation engine.

object	Transition model, object of class transitreg.
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.

Details

The function transforms the input data using transitreg_tmf() to a format compatible with transition models. Estimation relies on binary GLM-type techniques to model conditional transition probabilities. Note that the theta variable, representing the current transition level, must be included in the model by the user, please see the examples. Additional transition-specific covariates, such as those addressing excess zeros, can be included by adding theta0, theta1, etc., to the formula.

For continuous responses, the breaks argument specifies the intervals for discretizing the response, enabling the application of count-based transition models.

Value

An object of class "transitreg", which includes the following components:

- Fitted model results compatible with **mgcv**-style outputs.
- Methods for plot, summary, residuals, and predict.
- Model diagnostics and transformation details.

See transitreg_detect_cores() for some more details.

OpenMP

For improved performance "transitreg" is partially implemented in C, making use of the OpenMP library for parallelization. The argument ncores allows the user to control how many cores to be used for the C functions shipped with this package. By default, the number of cores will be detected automatically (ncores = NULL). The following rules are applied:

- If OpenMP is not available, the number of cores is set to 1L in all cases.
- If ncores = NULL the total number of cores (N) is detected automatically, and ncores is set to N - 2L.
- If ncores < 1L, one core will be used (single-core processing).
- If ncores > 1L all available cores will be utilized.

Author(s)

Niki

Reto

See Also

transitreg_tmf(), mgcv::gam()

Examples

```
## Example 1: Count data.
set.seed(123)
n <- 1000
x <- runif(n, -3, 3)
y <- rpois(n, exp(2 + sin(x)))

## Fit transition count response model.
b <- transitreg(y ~ s(theta) + s(x))

## GAM summary.
summary(b)

## GAM coefficients
coef(b)

## Effect plots.
plot(b)

## Plotting hanging rootogram, quantile residuals, and
## a probability integral transform (PIT) histogram.
plot(b, which = c("rootogram", "qqrplot", "pithist", "wormplot"))

## Predictions and plotting.
nd <- data.frame(x = seq(-3, 3, length = 100))
fit <- cbind(
  "97.5%"  = predict(b, nd, type = "quantile", p = 1 - 0.05/2),
  "median" = predict(b, nd, type = "quantile", p = 0.5),
  "2.5%"   = predict(b, nd, type = "quantile", p = 0.05/2),
  "mode"   = predict(b, nd, type = "mode"),
  "mean"   = predict(b, nd, type = "mean")
)

## Plot data and fitted counts.
plot(y ~ x, pch = 16, col = rgb(0.1, 0.1, 0.1, alpha = 0.4))
matplot(nd$x, fit, type = "l", add = TRUE, lwd = 2,
       col = c(4, 4, 4, 2, 3), lty = c(2, 1, 2, 1, 1))
legend("topleft", legend = colnames(fit),
       col = c(4, 4, 4, 2, 3), lty = c(2, 1, 2, 1, 1))

## Visualizing three predicted distributions via Transitreg distributions
idx <- c(25, 50, 75)
d3 <- Transition(predict(b, nd[idx, , drop = FALSE], type = "tp"), breaks = seq.int(0, b$bins))

par(ask = TRUE)
plot(d3)
abline(v = predict(b, nd[idx, , drop = FALSE], type = "mode"), col = 1:3, lty = 3)
plot(d3, tp = TRUE) # add transition probabilities
abline(v = predict(b, nd[idx, , drop = FALSE], type = "mode"), col = 1:3, lty = 3)

plot(d3, cdf = TRUE)
plot(d3, cdf = TRUE, tp = TRUE) # add transition probabilities
```

```
## Example 2: Continuous response.
set.seed(123)
n <- 1000
x <- runif(n, -3, 3)
y <- sin(x) + rnorm(n, sd = exp(-1 + cos(x)))

## Fit model with continuous response.
b <- transitreg(y ~ s(theta) + s(x) + te(x, theta), breaks = 200)

## Diagnostic plots
plot(b, which = c("rootogram", "qqrplot", "pithist", "wormplot"))

## Predictions and plotting
nd <- data.frame(x = seq(-3, 3, length = 100))
fit <- cbind(
  "97.5%"  = predict(b, nd, type = "quantile", p = 1 - 0.05/2),
  "median" = predict(b, nd, type = "quantile", p = 0.5),
  "2.5%"   = predict(b, nd, type = "quantile", p = 0.05/2),
  "mode"   = predict(b, nd, type = "mode")
)

## Plot data and fitted curves.
plot(y ~ x, pch = 16, col = rgb(0.1, 0.1, 0.1, alpha = 0.4))
matplot(nd$x, fit, type = "l", add = TRUE, lwd = 2,
       col = c(4, 4, 4, 2), lty = c(2, 1, 2, 1))
legend("topleft", legend = colnames(fit),
       col = c(4, 4, 4, 2), lty = c(2, 1, 2, 1))

## Visualizing 5 randomly selected fitted distributions
set.seed(6020)

par(ask = TRUE)
idx <- sample(seq_len(nrow(model.frame(b))), 5L)
plot(b[idx])
plot(b[idx], cdf = TRUE)
plot(b[idx], cdf = TRUE, tp = TRUE)


## Example 3: Count response with neural network.
set.seed(123)
n <- 1000
x <- runif(n, -3, 3)
y <- rpois(n, exp(2 + sin(x)))

## Fit NN transition count response model.
b <- transitreg(y ~ theta + x, engine = "nnet",
                size = 5, maxit = 1000, decay = 0.001)

## Predictions and plotting.
nd <- data.frame(x = seq(-3, 3, length = 100))
fit <- cbind(
```

```
  "97.5%"  = predict(b, nd, type = "quantile", p = 1 - 0.05/2),
  "median" = predict(b, nd, type = "quantile", p = 0.5),
  "2.5%"   = predict(b, nd, type = "quantile", p = 0.05/2),
  "mode"   = predict(b, nd, type = "mode")
)

## Plot data and fitted counts.
plot(y ~ x, pch = 16, col = rgb(0.1, 0.1, 0.1, alpha = 0.4))
matplot(nd$x, fit, type = "l", add = TRUE, lwd = 2,
       col = c(4, 4, 4, 2), lty = c(2, 1, 2, 1))
legend("topleft", legend = colnames(fit),
       col = c(4, 4, 4, 2), lty = c(2, 1, 2, 1))

## Visualizing 5 randomly selected fitted distributions
set.seed(2025)
idx <- sample(seq_len(nrow(model.frame(b))), 5L)

par(ask = TRUE)
plot(b[idx])
plot(b[idx], cdf = TRUE)
plot(b[idx], cdf = TRUE, tp = TRUE)
```

`transitreg_detect_cores` *Detect number of cores for OpenMP*

Description

The calculation of CDFs and PDFs is implemented in C and allows for parallelization using OpenMP. This function detects how may cores are available in total (if OpenMP is available).

Usage

```
transitreg_detect_cores(verbose = TRUE)
```

Arguments

`verbose`	logical, if `TRUE` a message is shown.

Value

Number of available cores (integer). If OpenMP is not available, `1L` is returned.

Author(s)

Reto

transitreg_get_number_of_cores *Get number of cores for OpenMP*

Description

Some parts of the package use C routines which allow for parallelization using OpenMP. This function is used to specify how many cores to be used.

Usage

```
transitreg_get_number_of_cores(ncores = NULL, verbose = FALSE)
```

Arguments

ncores	NULL or a positive integer.
verbose	logical, if TRUE a message is shown.

Details

If ncores is NULL the number of available cores is auto-detected and set to 'total number of cores - 2'. If integer, it is checked if this number of cores is available, else set tot he 'total number of cores available'.

Value

Number of cores to be used in OpenMP parallelization (integer).

Author(s)

Reto

transitreg_glmnet *Estimat Transition Probabilities via* glmnet

Description

Experimental engine used to estimate the transition probabilities in a transitreg() model.

Usage

```
transitreg_glmnet(formula, data, nfolds = 10, ...)

## S3 method for class 'transitreg_glmnet'
predict(object, type = "response", s = "lambda.min", ...)
```

Arguments

formula	An object of class formula.
data	A data.frame containing the required data to fit a binomial regression model (given formula) using glmnet::cv.glmnet().
nfolds	Integer, defaults to 10L. Number of cross-folds for the glmnet model.
...	currently unused.
object	an object of class transitreg_glmnet.
type	character, defaults to "response".
s	value(s) of the penalty parameter, defaults to "lambda.min". See glmnet::predict.glmnet() for details.

---

transitreg_tensorflow *Estimat Transition Probabilities via* tensorflow/keras3

Description

Experimental engine used to estimate the transition probabilities in a transitreg() model.

Usage

```
transitreg_tensorflow(
  formula,
  data,
  nlayers = 2,
  units = 20,
  epochs = 1000,
  batch_size = 16,
  activation = "relu",
  dropout = 0.1,
  validation_split = 0.2,
  patience = 10,
  trace = 0,
  ncores = 1,
  ...
)

## S3 method for class 'transitreg_tensorflow'
predict(object, newdata, ...)
```

Arguments

formula	An object of class formula.
data	A data.frame containing the required data to fit a tensorflow model (given formula) using keras3::keras_model_sequential().

nlayers	Number of layers.
units	... (TODO)
epochs	... (TODO)
batch_size	... (TODO)
activation	... (TODO)
dropout	... (TODO)
validation_split	... (TODO)
patience	... (TODO)
trace	... (TODO)
ncores	Number of cores to be used to train the model.
...	currently unused.
object	an object of class transitreg_tensorflow.
newdata	Data frame with the variables required for the prediction.

Author(s)

Niki

`transitreg_tmf` *Transition Model Data Preparer*

Description

Transforms a data frame into the format required for estimating transition models. The function generates binary response data for fitting GLM-type models, specifically designed to represent transitions between counts or states in a probabilistic framework.

Usage

```
transitreg_tmf(
  data,
  response,
  breaks,
  censored = c("uncensored", "left", "right", "both"),
  theta_vars = NULL,
  newresponse = NULL,
  scaler = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

data	A data frame containing the raw input data.
response	Character string specifying the name of the response variable to be used for transition modeling. This variable must represent counts or categorical states.
breaks	numeric vector with the bin intersection points (break points).
censored	character, or one of "uncensored" (default), "left", "right", or "both". Specifies if the distribution is uncensored or censored on one or both ends.
theta_vars	NULL or character vector with the bin-specific theta variables to be set up (e.g., c("theta0", "theta99")). Can be an empty character vector (or NULL) if there are no theta_vars.
newresponse	NULL or integer. New response vector to overwrite the one in data.
scaler	Can be FALSE/NULL, TRUE, or a list. See section 'Scaler' for details.
verbose	Logical value indicating whether information about the transformation process should be printed to the console. Default is FALSE.
...	currently unused (TODO(R): Check lower/upper dev option controlled via the dots argument; remove or document).

Details

Transition models focus on modeling the conditional probabilities of transitions between states or counts. This function converts the input data into a long format suitable for such models. Each row in the resulting data frame corresponds to a binary transition indicator, representing whether a transition to a higher category occurred. For details on the modeling framework, see Berger and Tutz (2021).

Value

A transformed data frame in the long format. Each row represents a binary transition indicator (Y) for the response variable. Additional columns in the output include:

- index: The original row index from the input data.
- Y: The binary indicator for whether a transition to a higher category occurred.
- theta: The level corresponding to the current transition.
- theta[0-9]+: Special binary variables which are 1L for all rows where (the unscaled) theta matches the integer [0-9]+$, else 0L. Indicator that we are in bin [0-9]+$. Only set if theta_vars is specified accordingly.

The return will contain the response name as attribute "response" and (potentially) a "scaler" attribute (see section 'Scaler').

This format is required for fitting transition models using GLM or GAM frameworks. For instance, a response variable with a value of 3 will generate rows with transitions up to its value (0, 1, 2, and 3).

'Scaler'

When estimating new model (transitreg()), the user can specify scale.x = TRUE which will standardize all covariates as well as the theta variable using (z - mean(z)) / sd(z) with z being a covariate or theta. If the input scaler to this function is set TRUE the return of this function will provide an attribute "scaler" which contains a list, storing mean and standard deviation used for scaling such that it can be applied to new data (e.g., during prediction). If FALSE or NULL no scaling is applied, and no attribute is set.

If input scaler is a list, the scaling is applied given the content of that list. No additional attribute will be set on the return value.

Author(s)

Niki

See Also

transitreg().

Examples

```
## Raw data frame.
d <- data.frame(myresponse = c(5.3, 0, 2.1),
                x = c(-1.2, 3.2, -0.5),
                y = c(765, 731, 353),
                z = as.factor(c("foo", "bar", "foo")))

## Building model.frame for testing
mf <- model.frame(myresponse ~ x + y + z, d)

## Breaks
bk <- seq(0, 8, by = 1)

## Simple case
tmf <- transitreg:::transitreg_tmf(mf, "myresponse", bk)
head(tmf)

## Providing response via 'newresponse' argument (bin index)
idx  <- transitreg:::num2bin(mf$myresponse, bk, "uncensored")
tmf2 <- transitreg:::transitreg_tmf(subset(mf, select = -myresponse),
                                    "myresponse", bk, newresponse = idx)
head(tmf2)

## Demonstrating 'thetaX' behavior
theta_vars <- c("theta0", "theta1", "theta2")
tmf3 <- transitreg:::transitreg_tmf(mf, "myresponse", bk, theta_vars = theta_vars)
head(tmf3)

## Demonstrating 'scaler' behavior
tmf4 <- transitreg:::transitreg_tmf(mf, "myresponse", bk, scaler = TRUE)
head(tmf4)
(scaler <- attr(tmf4, "scaler"))
```

```
## Re-using the scaler from above
tmf5 <- transitreg:::transitreg_tmf(mf, "myresponse", bk, scaler = scaler)
identical(tmf4, tmf5)
```

Index