Package 'durmod'

August 21, 2019

Type Package

Title Mixed Proportional Hazard Competing Risk Model

Version 1.1-2

Date 2019-08-21

URL https://github.com/sgaure/durmod

BugReports https://github.com/sgaure/durmod/issues

Maintainer Simen Gaure <simen@gaure.no>

Description
Estimation of piecewise constant mixed proportional hazard competing risk model with NPMLE. The model is described in S. Gaure et al. (2007) <doi:10.1016/j.jeconom.2007.01.015>, J. Heckman and B. Singer (1984) <doi:10.2307/1911491>, and B.G. Lindsay (1983) <doi:10.1214/aos/1176346059>.

Classification/JEL C14, C15, C41

License Artistic-2.0

Imports Rcpp (>= 1.0.1), stats, utils, numDeriv, nloptr, parallel, data.table, mvtnorm

Suggests knitr

Depends R (>= 3.5.0)

VignetteBuilder knitr

LinkingTo Rcpp

LazyData TRUE

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation yes

Author Simen Gaure [aut, cre] (<https://orcid.org/0000-0001-7251-8747>)

Repository CRAN

Date/Publication 2019-08-21 11:50:02 UTC

R topics documented:

`durmod-package` *A package for estimating a mixed proportional hazard competing risk model with the NPMLE.*

Description

The main method of the package is `mphcrm`. It has an interface somewhat similar to `lm`. There is an example of use in `datagen`, with a generated dataset similiar to the ones in *Gaure et al. (2007)*. For those who have used the program used in that paper, a mixture of R, Fortran, C, and python, this is an entirely new self-contained package, written from scratch with 12 years of experience. Currently not all functionality from that behemoth has been implemented, but most of it.

Details

A short description of the model follows.

There are some individuals with some observed covariates X_i. The individuals are observed for some time, so there is typically more than one observation of each individual. At any point they experience one or more hazards. The hazards are assumed to be of the form $h_i^j = exp(X_i\beta_j)$, where β_j are coefficients for hazard j. The hazards themselves are not observed, but an event associated with them is, i.e. a transition of some kind. The time of the transition, either exactly recorded, or within an interval, must also be in the data set. With enough observations it is then possible to estimate the coefficients β_j.

However, it just so happens that contrary to ordinary linear models, any unobserved heterogeneity may bias the estimates, not just increase uncertainty. To account for unobserved heterogeneity, a random intercept is introduced, so that the hazards are of the form $h_i^j(\mu_k) = exp(X_i\beta_j + \mu_k)$ for k between 1 and some n. The intercept may of course be written multiplicatively as $exp(X_i\beta_j)exp(\mu_k)$, that is why they are called *proportional* hazards.

The individual likelihood depends on the intercept, i.e. $L_i(\mu_k)$, but we integrate it out so that the individual likelihood becomes $\sum p_k L_i(\mu_k)$. The resulting mixture likelihood is maximized over all the βs, n, the μ_ks, and the probabilities p_k.

Besides the function `mphcrm` which does the actual estimation, there are functions for extracting the estimated mixture, they are `mphdist`, `mphmoments` and a few more.

There's a summary function for the fitted model, and there is a data set available with `data(durdata)` which is used for demonstration purposes. Also, an already fitted model is available there, as `fit`.

The package may use more than one cpu, the default is taken from `getOption("durmod.threads")` which is initialized from the environment variable `DURMOD_THREADS`, `OMP_THREAD_LIMIT`, `OMP_NUM_THREADS` or `NUMBER_OF_PROCESSORS`, or parallel::detectCores() upon loading the package.

For more demanding problems, a cluster of machines (from packages **parallel** or **snow**) can be used, in combination with the use of threads.

There is a vignette (`vignette("whatmph")`) with more details about **durmod** and data layout.

References

Gaure, S., K. Røed and T. Zhang (2007) *Time and causality: A Monte-Carlo Assessment of the timing-of-events approach*, Journal of Econometrics 141(2), 1159-1195. https://doi.org/10.1016/j.jeconom.2007.01.015

`a2p` *Convert probability parameters to probabilities*

Description

`mphcrm` parametrizes the probabilities that it optimizes. For $n+1$ probabilities there are n parameters a_j, such that probability $P_i = \frac{a_i}{\sum_j \exp(a_j)}$, where we assume that $a_0 = 0$.

Usage

```
a2p(a)

p2a(p)
```

Arguments

`a`	a vector of parameters
`p`	a vector of probabilities with sum(p) = 1

Value

`a2p` returns a vector probabilities with sum 1.

`p2a` returns a vector of parameters.

Examples

```
# Draw 5 parameters
a <- rnorm(5)
a
# make 6 probabilities
p <- a2p(a)
p
# convert back
p2a(p)
```

`datagen` *Generate example data*

Description

Generate a data table with example data

Usage

```
datagen(N, censor = 80)
```

Arguments

`N`	integer. The number of individuals in the dataset.
`censor`	numeric. The total observation period. Individuals are removed from the dataset if they do not exit to `"job"` before this time.

Details

The dataset simulates a labour market programme. People entering the dataset are without a job.

They experience two hazards, i.e. probabilities per time period. They can either get a job and exit from the dataset, or they can enter a labour market programme, e.g. a subsidised job or similar, and remain in the dataset and possibly get a job later. In the terms of this package, there are two transitions, `"job"` and `"program"`.

The two hazards are influenced by covariates observed by the researcher, called `"x1"` and `"x2"`. In addition there are unobserved characteristics influencing the hazards. Being on a programme also influences the hazard to get a job. In the generated dataset, being on a programme is the indicator variable `alpha`. While on a programme, the only transition that can be made is `"job"`.

The dataset is organized as a series of rows for each individual. Each row is a time period with constant covariates.

The length of the time period is in the covariate `duration`.

The transition being made at the end of the period is coded in the covariate `d`. This is an integer which is 0 if no transition occurs (e.g. if a covariate changes), it is 1 for the first transition, 2 for the second transition. It can also be a factor, in which case the level marking no transition must be called `"none"`.

The covariate alpha is zero when unemployed, and 1 if on a programme. It is used for two purposes. It is used as an explanatory variable for transition to job, this yields a coefficient which can be interpreted as the effect of being on the programme. It is also used as a "state variable", as an index into a "risk set". I.e. when estimating, the mphcrm function must be told which risks/hazards are present. When on a programme the "toprogram" transition can not be made. This is implemented by specifying a list of risksets and using alpha+1 as an index into this set.

The two hazards are modeled as $exp(X\beta + \mu)$, where X is a matrix of covariates β is a vector of coefficients to be estimated, and μ is an intercept. All of these quantities are transition specific. This yields an individual likelihood which we call $M_i(\mu)$. The idea behind the mixed proportional hazard model is to model the individual heterogeneity as a probability distribution of intercepts. We obtain the individual likelihood $L_i = \sum_j p_j M_i(\mu_j)$, and, thus, the likelihood $L = \sum_j L_j$.

The likelihood is to be maximized over the parameter vectors β (one for each transition), the mass-points μ_j, and probabilites p_j.

The probability distribution is built up in steps. We start with a single masspoint, with probability 1. Then we search for another point with a small probability, and maximize the likelihood from there. We continue with adding masspoints until we no longer can improve the likelihood.

Note

The example illustrates how data(durdata) was generated.

Examples

```
data.table::setDTthreads(1)  # avoid screams from cran-testing
dataset <- datagen(5000,80)
print(dataset)
risksets <- list(unemp=c("job","program"), onprogram="job")
# just two iterations to save time
Fit <- mphcrm(d ~ x1+x2 + ID(id) + D(duration) + S(alpha+1) + C(job,alpha),
          data=dataset, risksets=risksets,
          control=mphcrm.control(threads=1,iters=2))
best <- Fit[[1]]
print(best)
summary(best)
```

---

durdata *Duration data*

Description

The dataset simulates a labour market programme. People entering the dataset are without a job.

Usage

```
data(durdata)
```

Format

A data.frame

Details

They experience two hazards, i.e. probabilities per time period. They can either get a job and exit from the dataset, or they can enter a labour market programme, e.g. a subsidised job or similar, and remain in the dataset and possibly get a job later. In the terms of this package, there are two transitions, `"job"` and `"program"`.

The two hazards are influenced by covariates observed by the researcher, called `"x1"` and `"x2"`. In addition there are unobserved characteristics influencing the hazards. Being on a programme also influences the hazard to get a job. In the generated dataset, being on a programme is the indicator variable `alpha`. While on a programme, the only transition you can make is `"job"`.

The dataset is organized as a series of rows for each individual. Each row is a time period with constant covariates.

The length of the time period is in the covariate `duration`.

The transition being made at the end of the period is coded in the covariate `d`. This is an integer which is 0 if no transition occurs (e.g. if a covariate changes), it is 1 for the first transition, 2 for the second transition. It can also be a factor, in which case the level marking no transition must be called `"none"`. In the test dataset it is a factor with the levels `"none"`, `"job"`, and `"program"`.

The covariate `alpha` is zero when unemployed, and 1 if on a programme. It is used for two purposes. It is used as an explanatory variable for transition to job, this yields a coefficient which can be interpreted as the effect of being on the programme. It is also used as a "state variable", as an index into a "risk set". I.e. when estimating, the `mphcrm` function must be told which risks/hazards are present. When on a programme the `"toprogram"` transition can not be made. This is implemented by specifying a list of risksets and using `alpha+1` as an index into this set.

The dataset has already been fitted in the `fit` object.

Examples

```
data(durdata)
print(durdata)
print(fit)
summary(fit[[1]])
```

`flatten` *Convert a structured coefficient set to a vector*

Description

`mphcrm` stores coefficients in a list, not in a vector. This is because they should be treated differently according to whether they are probabilities, proportional hazards, or coefficients for factor levels or ordinary covariates. `flatten` extracts them as a named vector. `unflatten` puts them back in structured form.

Usage

```
flatten(x, exclude = attr(x, "exclude"))

unflatten(flesh, skeleton = attr(flesh, "skeleton"),
  exclude = attr(flesh, "exclude"))
```

Arguments

x	parameter set as typically found in opt[[1]]$par, where opt is returned from mphcrm.
exclude	For internal use
flesh	vector of class "relistable", as returned from flatten.
skeleton	For internal use

Details

flatten/unflatten is just a thinly disguised unlist/relist, but uses slightly more readable names.

---

geninv *Moore-Penrose generalized inverse*

Description

Moore-Penrose generalized inverse

Usage

```
geninv(X, tol = .Machine$double.eps^(2/3))
```

Arguments

X	matrix
tol	tolerance for determining bad entries

Value

A matrix of the same dimension as X is returned, the Moore-Penrose generalized inverse.

Examples

```
# create a positive definite 5x5 matrix
x <- crossprod(matrix(rnorm(25),5))
# make it singular
x[,2] <- x[,3]+x[,5]
geninv(x)
```

mphcrm *Estimate a mixed proportional hazard model*

Description

mphcrm implements estimation of a mixed proportional hazard competing risk model. The baseline hazard is of the form $exp(X\beta)$ where X is a matrix of covariates, and β is a vector of parameters to estimate. In addition there is an intercept term μ, i.e. the hazard is $exp(X\beta + \mu)$. There are several transitions to be made, and a set of X, β, and μ for each possible transition.

Each individual may have several observations, with either a transition at the end of the observation, or not a transition. It is a competing risk, there can be more than one possible transition for an observation, but only one is taken at the end of the period.

For each individual i there is a log likelihood as a function of μ, called $M_i(\mu)$.

The mixture part is that the μ's are stochastic. I.e. we have probabilities p_j, and a vector of μ_j of masspoints (one for each transition), for each such j.

So the full likelihood for an individual is $L_i = \sum_j p_j M_i(\mu_j)$.

The mphcrm() function maximizes the likelihood $\sum_i L_i$ over p_j, mu_j, and β.

In addition to the covariates specified by a formula, a variable which records the duration of each observation must be specified.

In some datasets it is known that not all risks are present at all times. Like, losing your job when you do not have one. In this case it should be specified which risks are present.

The estimation starts out with one masspoint, maximizes the likelihood, tries to add another point, and continues in this fashion.

Usage

```
mphcrm(formula, data, risksets = NULL, timing = c("exact", "interval",
  "none"), subset, na.action, control = mphcrm.control())
```

Arguments

formula	A formula specifying the covariates. In a formula like d ~ x1 + x2 + ID(id) + D(dur) + C(job,alpha) + S(state), the d is the transition which is taken, coded as an integer where 0 means no transition, and otherwise d is the number of the transition which is taken. d can also be a factor, in which case the level which is no transition must be named "0" or "none". If d is an integer, the levels for transitions will be named "t1", "t2", ... And "none". The x1+x2 part is like in lm, i.e. ordinary covariates or factors. The D() specifies the covariate which holds the duration of each observation. The transition in d is assumed to be taken at the end of this period. The ID() part specifies the covariate which holds the individual identification. The S() specifies the covariate which holds an index into the risksets list. These three special symbols are replaced with I(), so it is possible to have calculations inside them.

If the covariates differ among the transitions, one can specify covariates conditional on the transition taken. If e.g. the covariates `alpha` and `x3` should only explain transition to job, specify `C(job,alpha+x3)`. This comes in addition to the ordinary covariates. Then name `job` refers to a level in `d`, the transition taken.

`data` A data frame which contains the covariates. It must be sorted on individuals.

`risksets` A list of character vectors. Each vector is a list of transitions, i.e. which risks are present for the observation. The elements of the vectors must be levels of the covariate which is the left hand side of the `formula`. If the state variable in the formula is a factor, the `risksets` argument should be a named list, with names matching the levels of `state`. If all risks are present at all times, the `risksets`-argument can be specified as `NULL`, or ignored.

`timing` character. The timing in the duration model. Can be one of

- `"exact"`. The timing is exact, the transition occured at the end of the observation interval.
- `"interval"`. The transition occured some time during the observation interval. This model can be notoriously hard to estimate due to unfavourable numerics.
- `"none"`. There is no timing, the transition occured, or not. A logit model is used.

`subset` For specifying a subset of the dataset, similar to `lm`.

`na.action` For handling of missing cases, similar to `lm`.

`control` List of control parameters for the estimation. See `mphcrm.control`.

Details

The estimation starts by estimating the null-model, i.e. all parameters set to 0, only one intercept for each transition is estimated.

Then it estimates the full model, still with one intercept in each transition.

After the initial model has been estimated, it tries to add a masspoint to the mixing distribution, then estimates the model with this new distribution.

The algorithm continues to add masspoints in this way until either it can not improve the likelihood, or the number of iterations as specified in `control$iters` are reached.

The result of every iteration is returned in a list.

If you interrupt `mphcrm` it will catch the interrupt and return with the estimates it has found so far. This behaviour can be switched off with `control$trap.interrupt=FALSE`.

Value

A list, one entry for each iteration. Ordered in reverse order. Ordinarily you will be interested in the first entry.

Note

The algorithm is not fully deterministic. New points are searched for randomly, there is no canonical order in which they can be found. It can happen that a point is found early which makes the rest of the estimation hard, so it terminates early. In particular when using interval timing. One should then make a couple of runs to ensure they yield reasonably equal results.

See Also

A description of the dataset is available in datagen and durdata, and in the vignette `vignette("whatmph")`

Examples

```
data(durdata)
head(durdata)
risksets <- list(c('job','program'), c('job'))
Fit <- mphcrm(d ~ x1+x2 + C(job,alpha) + ID(id) + D(duration) + S(alpha+1), data=durdata,
     risksets=risksets, control=mphcrm.control(threads=1,iters=2))
best <- Fit[[1]]
summary(best)
```

---

mphcrm.callback *Default callback function for mphcrm*

Description

The default callback function prints a line whenever estimation with a masspoint is completed.

Usage

```
mphcrm.callback(fromwhere, opt, dataset, control, ...)
```

Arguments

fromwhere	a string which identifies which step in the algorithm it is called from. `fromwhere=='full'` means that it is a full estimation of all the parameters. There are also other codes, when adding a point, when removing duplicate points. When some optimization is completed it is called with the return status from optim (and in some occasions from nloptr).
opt	Typically the result of a call to optim.
dataset	The dataset in a structured form.
control	The control argument given to mphcrm
...	other arguments

Details

If you write your own callback function it will replace the default function, but you can of course call the default callback from your own callback function, and in addition print your own diagnostics, or save the intermediate `opt` in a file, or whatever. You can even stop the estimation by doing a `stop('<some message>')`, and `mphcrm` will return with the estimates done so far, provided the control parameter `trap.interrupt=TRUE`.

Note

Beware that `control` contains a reference to the callback function, which may contain a reference to the top-level environment, which may contain the full dataset. So if you save `control` to file, you may end up saving the entire dataset.

Examples

```
callback <- function(fromwhere, opt, dataset, control, ...) {
  # call the standard callback to print a diagnostic line
  mphcrm.callback(fromwhere, opt, dataset, control, ...)
  # print the distribution and two coefficients
  if(fromwhere == 'full') {
    print(round(mphdist(opt),6))
    print(summary(opt)$coefs[c('job.alpha','job.x1'),])
  }
}
```

---

mphcrm.control *Control parameters for mphcrm*

Description

Modify the default estimation parameters for `mphcrm`.

Usage

```
mphcrm.control(...)
```

Arguments

`...`	parameters that can be adjusted. See the `vignette("whatmph")` for more details. • threads. integer. The number of threads to use. Defaults to `getOption('durmod.threads')`. • iters. integer. How many iterations should we maximally run. Defaults to 12. • ll.improve. numeric. How much must the log-likelihood improve from the last iteration before termination. Defaults to 0.001. • newpoint.maxtime. numeric. For how many seconds should a global search for a new point improving the likelihood be conducted before we continue with the best we have found. Defaults to 120.

- callback. A user-specified `function(fromwhere,opt,dataset,control,...)` which is called after each optimization step. It can be used to report what is happening, check whatever it wants, and optionally stop the estimation by calling stop(). In this case, `mphcrm()` will return with the most recently estimated set of parameters. See the help on `mphcrm.callback` for information on the argument.
- trap.interrupt. logical. Should interrupts be trapped so that `mphcrm` returns gracefully? In this case the program will continue. Defaults to `interactive()`.
- cluster. Cluster specification from package **parallel** or **snow**.

Value

List of control parameters suitable for the `control` argument of `mphcrm`.

Note

There are more parameters documented in the `vignette("whatmph")`. Some of them can be useful. Instead of cluttering the source code with constants and stuff required by various optimization routines, they have been put in this control list.

`mphdist` *Extract the mixed proportional hazard distribution*

Description

Various functions for extracting the proportional hazard distribution.

`mphdist` extracts the hazard distribution.

`mphdist.log` extracts the log hazard distribution.

`mphmoments` returns the first and second moments of the hazard distribution.

`mphmoments.log` returns the first and second moments of the log hazard distribution.

`mphcov` returns the variance/covariance matrix of the hazard distribution.

`mphmedian` returns the medians of the hazard distribution.

`mphcov.log` returns the variance/covariance matrix of the log hazard distribution.

Usage

```
mphdist(pset)

mphdist.log(pset)

mphmoments(pset)

mphmoments.log(pset)

mphcov(pset)
```

```
mphmedian(pset)

mphcov.log(pset)
```

Arguments

pset	a parameter set of class "mphcrm.pset", typically opt[[1]]$par, where opt is returned from mphcrm. If given a list of results, extracts the first in the list.

Value

A matrix.

Examples

```
# load a dataset and a precomputed fitted model
data(durdata)
best <- fit[[1]]
mphdist(best)
mphmoments(best)
mphcov.log(best)
```

---

pseudoR2 *Calculate various pseudo R2's*

Description

There are several variants of pseudo R^2 that can be computed for a likelihood estimation. They all relate the log likelihood of the estimated model to the log likelihood of the null model.

The ones included here are McFadden's, Adjusted McFadden's, Cox & Snell's, and Nagelkerke, Cragg, and Uhler's.

Usage

```
pseudoR2(opt)
```

Arguments

opt	returned value from mphcrm.

Value

A matrix is returned, with one row for each iteration containing the various pseudo R^2s.

se *Extract standard errors of the estimated parameters*

Description

Extract standard errors of the estimated parameters

Usage

```
se(x, tol = .Machine$double.eps)
```

Arguments

x	The Fisher matrix, typically from `opt[[1]]$fisher`, where `opt` is returned from `mphcrm`.
tol	tolerance for geninv

Value

A named vector of standard errors is returned.

smashlevels *Collapse levels of a factor*

Description

Combines levels of a factor into new levels

Usage

```
smashlevels(f, newlevels)
```

Arguments

f	factor.
newlevels	list. The names of `newlevels` are the new levels. Each list element is a list of old levels in the factor `f` which should be combined into the new level

Examples

```
# create a factor with levels 30:60
f <- factor(sample(30:60, 200, replace=TRUE))
# combine 35-40 into a single level, 41-50 into a single level, and 51-60 into a single level
g <- smashlevels(f, list(`35-40` = 35:40, `41-50` = 41:50, `51-60` = 51:60))
table(g)
# If the syntax permits, the backticks can be avoided.
h <- smashlevels(f, list(young=30:34, pushing40 = 35:40, pushing50 = 41:50, fossilized = 51:120))
table(h)
```

`timestr` *Prettyprint a time interval*

Description

Converts a time in seconds to a short string e.g. `"3m4s"`.

Usage

```
timestr(t)
```

Arguments

`t`	numeric. time in seconds.

Value

A character string is returned.

Examples

```
timestr(1.3)
timestr(73)
timestr(4684)
```

Index