Package 'rstanjm'

October 28, 2016

Type Package

Title Bayesian joint longitudinal and time-to-event models via Stan

Version 0.0.0-1

Date 2016-06-28

Description Estimates joint longitudinal and time-to-event models using the 'rstan' package, which provides the R interface to the Stan C++ library for Bayesian estimation. Users specify models via the customary R syntax with formulas and data frames plus some additional arguments for priors.

License GPL (>=3)

Depends R (>= 3.0.2), Rcpp (>= 0.11.0), methods, survival (>= 2.39.4), lme4 (>= 1.1-8),

Imports rstanarm (>= 2.11.1), data.table (>= 1.9.6), ggplot2 (>= 2.0.0), cowplot (>= 0.6.2), Matrix (>= 1.2.6), nlme (>= 3.1-124), rstan (>= 2.9.0-3), shinystan (>= 2.1.0), stats, utils

Suggests gridExtra, HSAUR3, KernSmooth, knitr, MASS, rmarkdown, roxygen2, testthat

LinkingTo StanHeaders (>= 2.9.0), rstan (>= 2.9.0-3), BH (>= 1.58.0), Rcpp (>= 0.11.0), RcppEigen

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

RcppModules stan_fit4jm_mod

RoxygenNote 5.0.1

Author Sam Brilleman [cre, aut, cph],
Jonah Gabry [aut],
Ben Goodrich [aut],
Trustees of Columbia University [cph]

Maintainer Sam Brilleman <`sam.brilleman@monash.edu`>

Archs x64

R topics documented:

`rstanjm-package` *Bayesian joint longitudinal and time-to-event models via Stan*

Description

The **rstanjm** package allows the user to fit "joint models" for longitudinal and time-to-event data (also known as shared parameter models) under a Bayesian framework. Both univariate (one longitudinal marker) and multivariate (more than one longitudinal marker) joint models are allowed. Both continuous and non-continuous (e.g. binary or count data) longitudinal markers can be accomodated through a range of possible link functions and error distributions. Multi-level clustered data are allowed (e.g. patients within clinics), provided that the individual (e.g. patient) is the lowest level of clustering. Competing risk events are not currenly allowed.

For each longitudinal submodel, a generalised linear mixed model is assumed with one of the family and link combinations allowed by the **lme4** package.

If a multivariate joint model is specified (i.e. more than one longitudinal marker) then dependence between the different longitudinal markers is captured through correlated random effects; that is, a multivariate normal distribution with an unstructured correlation matrix is assumed for all random effects within a given clustering level (e.g. individual-level random effects).

If a model with multi-level clustering (i.e. clustering beyond the level of the individual) is specified, then a different correlation matrix is estimated for each clustering level.

For the time-to-event submodel a parametric proportional hazards model is assumed. The baseline hazard can be estimated using either a Weibull distribution or approximated using restricted cubic splines. Time-varying covariates are allowed in both the longitudinal and event submodels. Competing risk events are not currently allowed. The association structure for the joint model can be based on any of the following parameterisations: current value of the linear predictor in the longitudinal submodel(s); current expected value in the longitudinal submodel(s); first derivative (slope) of the linear predictor in the longitudinal submodel(s); shared random effects; no association structure (equivalent to fitting separate longitudinal and survival models).

Estimation of the joint model is under a Bayesian framework using Markov chain Monte Carlo (MCMC) methods, with the underlying estimation implemented using Stan (`http://mc-stan.org/`). A variety of prior distributions are available for the regression coefficients, including normal, Cauchy and t distributions, as well as shrinkage priors. The prior distribution for the covariance matrix of the individual-level (or other clustering-level) random effects is based on a novel decomposition. All prior distributions are implemented via the **rstanarm** package; see `priors` for details.

Author(s)

Sam Brilleman <`<sam.brilleman@monash.edu>`>.

Whilst acknowledging this package draws heavily on code from the **rstanarm** package, of which I am not an author.

References

Stan Development Team. (2015). *Stan Modeling Language Users Guide and Reference Manual.* `http://mc-stan.org/documentation/`

See Also

`stan_jm` for the main modelling function, as well as `stanjm-object`, `stanjm-methods`, `posterior_traj`, `posterior_survfit`, `posterior_predict`.

Also see `http://mc-stan.org/` for more information on the Stan C++ package used for model fitting.

`as.matrix.stanjm` *Extract the posterior sample*

Description

For models fit using MCMC (`algorithm="sampling"`), the posterior sample —the post-warmup draws from the posterior distribution— can be extracted from a fitted model object as a matrix, data frame, or array. The `as.matrix` and `as.data.frame` methods merge all chains together, whereas the `as.array` method keeps the chains separate.

Usage

```
## S3 method for class 'stanjm'
as.matrix(x, ..., pars = NULL, regex_pars = NULL)

## S3 method for class 'stanjm'
as.array(x, ..., pars = NULL, regex_pars = NULL)

## S3 method for class 'stanjm'
as.data.frame(x, ..., pars = NULL, regex_pars = NULL)
```

Arguments

x	A fitted model object returned by the stan_jm modelling function. See stanjm-object.
...	Ignored.
pars	An optional character vector of parameter names.
regex_pars	An optional character vector of regular expressions to use for parameter selection. regex_pars can be used in place of pars or in addition to pars.

Value

A matrix, data.frame, or array, the dimensions of which depend on pars and regex_pars as well as the model.

See Also

stanjm-methods

---

examplejm *Example joint longitudinal and time-to-event model*

Description

A model for use in **rstanjm** examples.

Format

Calling example("examplejm") will run the model in the Examples section, below, and the resulting stanjm object will then be available in the global environment. The chains and iter arguments are specified to make this example be small in size. In practice, we recommend that they be left unspecified in order to use the default values (4 and 2000 respectively) or increased if there are convergence problems. The cores argument is optional and on a multicore system, the user may well want to set that equal to the number of chains being executed.

Examples

```
examplejm <-
  stan_jm(formulaLong = logBili ~ year + (1 | id),
          dataLong = pbcLong,
          formulaEvent = Surv(futimeYears, death) ~ sex + trt,
          dataEvent = pbcSurv,
          time_var = "year",
          chains = 1, cores = 1, seed = 12345,
          iter = 1500, warmup = 750)
```

`pairs.stanjm` *Pairs method for stanjm objects*

Description

See `pairs.stanfit` for details.

Usage

```
## S3 method for class 'stanjm'
pairs(x, ...)
```

Arguments

`x`	A stanjm object.
`...`	Arguments to pass to `pairs.stanfit`.

Details

See the Details section in `pairs.stanfit`.

Examples

```
if (!exists("examplejm")) example(examplejm)
pairs(examplejm, pars = c("(Intercept)", "log-posterior"))
```

`plot.predict.stanjm` *Plot estimated subject-specific longitudinal trajectory*

Description

Various plots comparing the observed outcome variable from one of the longitudinal submodels y to simulated datasets y^{rep} from the posterior predictive distribution. This function is modelled on the `pp_check` function from the **rstanarm** package.

Usage

```
## S3 method for class 'predict.stanjm'
plot(x, ids = NULL, limits = c("ci", "pi", "none"),
  xlab = NULL, ylab = NULL, abline = FALSE, plot_observed = FALSE,
  facet_scales = "free_x", ci_geom_args = NULL, ...)
```

Arguments

`x`	A data frame and x of class `predict.stanjm` returned by a call to the function `posterior_traj`. The x contains point estimates and uncertainty interval limits for fitted values of the longitudinal response.
`ids`	An optional vector providing a subset of IDs for whom the predicted curves should be plotted.
`limits`	A quoted character string specifying the type of limits to include in the plot. Can be one of: `"ci"` for the Bayesian posterior uncertainty interval (often known as a credible interval); `"pi"` for the prediction interval; or `"none"` for no interval limits.
`xlab, ylab`	An optional axis label passed to `labs`.
`abline`	A logical. If `TRUE` then a vertical dashed line is added to the plot indicating the event or censoring time for the individual. Can only be used if each plot within the figure is for a single individual.
`plot_observed`	A logical. If `TRUE` then the observed longitudinal measurements are overlaid on the plot.
`facet_scales`	A character string passed to the `scales` argument of `facet_wrap` when plotting the longitudinal trajectory for more than one individual.
`ci_geom_args`	Optional arguments passed to `geom_ribbon` and used to control features of the plotted interval limits.
`...`	Optional arguments passed to `geom_smooth` and used to control features of the plotted trajectory.

Value

A `ggplot` x, also of class `plot.predict.stanjm`. This x can be further customised using the **ggplot2** package. It can also be passed to the function `plot_stack`.

See Also

`posterior_traj`, `plot_stack`

`plot.survfit.stanjm` *Plot estimated subject-specific or marginal survival function*

Description

Various plots comparing the observed outcome variable from one of the longitudinal submodels y to simulated datasets y^{rep} from the posterior predictive distribution. This function is modelled on the `pp_check` function from the **rstanarm** package.

Usage

```
## S3 method for class 'survfit.stanjm'
plot(x, ids = NULL, limits = c("ci", "none"),
  xlab = NULL, ylab = NULL, facet_scales = "free", ci_geom_args = NULL,
  ...)
```

Arguments

`x`	A data frame and x of class `survfit.stanjm` returned by a call to the function `posterior_survfit`. The x contains point estimates and uncertainty interval limits for estimated values of the survival function.
`ids`	An optional vector providing a subset of IDs for whom the predicted curves should be plotted.
`limits`	A quoted character string specifying the type of limits to include in the plot. Can be one of: `"ci"` for the Bayesian posterior uncertainty interval (often known as a credible interval); or `"none"` for no interval limits.
`xlab, ylab`	An optional axis label passed to `labs`.
`facet_scales`	A character string passed to the `scales` argument of `facet_wrap` when plotting the longitudinal trajectory for more than one individual.
`ci_geom_args`	Optional arguments passed to `geom_ribbon` and used to control features of the plotted interval limits.
`...`	Optional arguments passed to `geom_line` and used to control features of the plotted trajectory.

Value

A `ggplot` x, also of class `plot.survfit.stanjm`. This x can be further customised using the **ggplot2** package. It can also be passed to the function `plot_stack`.

See Also

`posterior_survfit`, `plot_stack`

`plot_stack` *Combine plots of estimated longitudinal trajectories and survival function*

Description

This function takes plots of estimated subject-specific longitudinal trajectories and the estimated subject-specific survival function and combines them into a single figure.

Usage

```
plot_stack(yplot, survplot)
```

Arguments

`yplot`	An object of class `plot.predict.stanjm`, returned by a call to the plotting method for objects of class `predict.stanjm`. If there is more than one longitudinal outcome, then a list of such objects can be provided.
`survplot`	An object of class `plot.survfit.stanjm`, returned by a call to the plotting method for objects of class `survfit.stanjm`.

Value

A single `ggplot` object that includes plots of the estimated subject-specific longitudinal trajectories stacked on top of their associated subject-specific survival curves.

`posterior_interval` *Posterior uncertainty intervals*

Description

The `posterior_interval` function computes Bayesian posterior uncertainty intervals for the model parameters, via a call to the `posterior_interval` function in the **rstanarm** package. See `posterior_interval` for details.

Usage

```
posterior_interval(object, ...)
```

Arguments

`object`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`...`	Arguments passed to the `posterior_interval` function of the **rstanarm** package.

Value

A matrix with two columns and as many rows as model parameters (or the subset of parameters specified by `pars` and/or `regex_pars`). For a given value of `prob`, p, the columns correspond to the lower and upper $100p\%$ interval limits and have the names $100\alpha/2\%$ and $100(1-\alpha/2)\%$, where $\alpha = 1 - p$. For example, if `prob=0.9` is specified (a 90% interval), then the column names will be `"5%"` and `"95%"`, respectively.

References

Gelman, A. and Carlin, J. (2014). Beyond power calculations: assessing Type S (sign) and Type M (magnitude) errors. *Perspectives on Psychological Science*. 9(6), 641–51.

Morey, R. D., Hoekstra, R., Rouder, J., Lee, M. D., and Wagenmakers, E. (2015). The fallacy of placing confidence in confidence intervals. *Psychonomic Bulletin & Review*. 1–21.

Examples

```
if (!exists("examplejm")) example(examplejm)
posterior_interval(examplejm)
posterior_interval(examplejm, regex_pars = "Long1")
posterior_interval(examplejm, regex_pars = c("Event", "Assoc"))
posterior_interval(examplejm, regexpars = "b[", prob = 0.95)
posterior_interval(examplejm, pars = "Long1|(Intercept)", prob = 0.5)
```

`posterior_predict` *Draw from posterior predictive distribution for the longitudinal submodel*

Description

The posterior predictive distribution is the distribution of the outcome implied by the model after using the observed data to update our beliefs about the unknown parameters in the model. Simulating data from the posterior predictive distribution using the observed predictors is useful for checking the fit of the model. Drawing from the posterior predictive distribution at interesting values of the predictors also lets us visualize how a manipulation of a predictor affects (a function of) the outcome(s). With new observations of predictor variables we can use the posterior predictive distribution to generate predicted outcomes.

Usage

```
posterior_predict(object, m = 1, newdata = NULL, re.form = NULL,
  fun = NULL, draws = NULL, seed = NULL, offset = NULL, ...)
```

Arguments

`object`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`m`	Integer specifying the number of the longitudinal submodel
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the model matrix is used. If `newdata` is provided and any variables were transformed (e.g. rescaled) in the data used to fit the model, then these variables must also be transformed in `newdata`. This only applies if variables were transformed before passing the data to one of the modeling functions and *not* if transformations were specified inside the model formula. Also see the Note section below for a note about using the `newdata` argument with with binomial models.
`re.form`	If `object` contains `group-level` parameters, a formula indicating which group-level parameters to condition on when making predictions. `re.form` is specified in the same form as for `predict.merMod`. The default, `NULL`, indicates that all estimated group-level parameters are conditioned on. To refrain from conditioning on any group-level parameters, specify `NA` or `~0`. The `newdata` argument may include new *levels* of the grouping factors that were specified when the model was estimated, in which case the resulting posterior predictions marginalize over the relevant variables.
`fun`	An optional function to apply to the results. `fun` is found by a call to `match.fun` and so can be specified as a function object, a string naming a function, etc.
`draws`	An integer indicating the number of draws to return. The default and maximum number of draws is the size of the posterior sample.
`seed`	An optional `seed` to use.
`offset`	A vector of offsets. Only required if `newdata` is specified and an `offset` argument was specified when fitting the model.
`...`	Currently unused.

Value

A `draws` by `nrow(newdata)` matrix of simulations from the posterior predictive distribution. Each row of the matrix is a vector of predictions generated using a single draw of the model parameters from the posterior distribution.

Note

For binomial models with a number of trials greater than one (i.e., not Bernoulli models), if `newdata` is specified then it must include all variables needed for computing the number of binomial trials to use for the predictions. For example if the left-hand side of the model formula is `cbind(successes, failures)` then both `successes` and `failures` must be in `newdata`. The particular values of `successes` and `failures` in `newdata` do not matter so long as their sum is the desired number of trials. If the left-hand side of the model formula were `cbind(successes, trials - successes)` then both `trials` and `successes` would need to be in `newdata`, probably with `successes` set to `0` and `trials` specifying the number of trials. See the Examples section below and the *How to Use the rstanarm Package* for examples.

See Also

`pp_check` for graphical posterior predictive checks of the longitudinal submodel(s).

`posterior_survfit` *Estimate the marginal or subject-specific survival function*

Description

The posterior predictive distribution is the distribution of the outcome implied by the model after using the observed data to update our beliefs about the unknown parameters in the model. Simulating data from the posterior predictive distribution using the observed predictors is useful for checking the fit of the model. Drawing from the posterior predictive distribution at interesting values of the predictors also lets us visualize how a manipulation of a predictor affects (a function of) the outcome(s). With new observations of predictor variables we can use the posterior predictive distribution to generate predicted outcomes.

Usage

```
posterior_survfit(object, newdataEvent = NULL, newdataLong = NULL, ids,
  times = NULL, limits = c(0.025, 0.975), condition = TRUE,
  extrapolate = TRUE, extrapolate_args = list(dist = NULL, prop = 0.5,
  increments = 15), marginalised = FALSE, draws = NULL, fun = NULL,
  seed = NULL, ...)
```

Arguments

`object`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`newdataLong, newdataEvent`	Optionally, new data frames in which to look for variables with which to predict. The data should be provided in the same format as for the original call to the estimation function `stan_jm`.

`ids`	An optional character vector containing the IDs of individuals for whom the estimated survival probabilites should be obtained. This defaults to `NULL` which returns predictions for all individuals in the original model if `newdataEvent` is `NULL`, or otherwise all individuals in `newdataEvent`.
`times`	An optional scalar or numeric vector specifying the values of `time_var` in the original model at which to obtain the estimated survival probabilities. If not specified, and `marginalised = FALSE`, then the default behaviour is to use the latest observation time for each individual (taken to be the event or censoring time if `newdata` is not specified, or the lastest value of `time_var` if `newdata` is specified) and extrapolate forward from there. If not specified, and `marginalised = TRUE`, then the default behaviour is to calculate the marginal survival probability at `n_increments` time points between baseline and the latest observation or event/censoring time for all individuals.
`limits`	A numeric vector of length 2 specifying the limits to use for the credible interval.
`condition`	A logical specifying whether the estimated subject-specific survival probabilities at time `t` should be conditioned on survival up to a fixed time point `u`. The default is to condition on the latest observation time for each individual (taken to be the event or censoring time if `newdata` is not specified, or the lastest value of `time_var` if `newdata` is specified). It cannot be set to `TRUE` if `marginalised` is set to `TRUE`.
`extrapolate`	A logical specifying whether to extrapolate the estimated survival function beyond the times specified in the `times` argument If `TRUE` then the extrapolation can be further controlled using the `extrapolate_args` argument.
`extrapolate_args`	A named list with parameters controlling extrapolation of the estimated survival function when `extrapolate = TRUE`. The list can contain the following named elements: `dist`, a positive scalar specifying the amount of time across which to forecast the estimated survival function; `prop`, a positive scalar between 0 and 1 specifying the amount of time across which to forecast the estimated survival function but represented as a proportion of the total observed follow up time for each individual; `increments`, a positive integer specifying the number of increments in time at which to calculate the forecasted survival probabilities. See the **Examples** below.
`marginalised`	A logical specifying whether the estimated subject-specific survival probabilities should be averaged (marginalised) across all individuals for whom the predictions were obtained. This can be used to obtain an estimate of the marginal survival probability. It cannot be set to `TRUE` if `condition` is set to `TRUE`.
`draws`	An integer indicating the number of MCMC draws to return. The default and maximum number of draws is the size of the posterior sample.
`fun`	An optional function to apply to the results. `fun` is found by a call to `match.fun` and so can be specified as a function object, a string naming a function, etc.
`seed`	An optional `seed` to use.
`...`	Currently unused.

Value

A named list with two elements, 'times' and 'survprobs'. The former is a list of times at which the survival probabilities are calculated. The latter is a list containing the correponding survival probabilities. Each element of 'survprobs' contains a `draws` by `levels(ids)` matrix of estimated survival probabilities. Each row of the matrix is a vector of predictions generated using a single draw of the model parameters from the posterior distribution.

See Also

ps_check for for graphical checks of the estimated marginal survival function, and posterior_predict for drawing from the posterior predictive distribution for the longitudinal submodel.

Examples

```
The numeric value should represent the proportion of observed follow up time
  (difference between baseline and \code{time}), for example, 0.2 = 20% of the
  observed follow up time. If set to \code{NULL} then survival probabilities
  are only calculated at \code{time} and no extrapolation is carried out.
  It is ignored if \code{marginalised} is set to \code{TRUE}.
```

`posterior_traj` *Estimate the marginal or subject-specific longitudinal trajectory*

Description

The posterior predictive distribution is the distribution of the outcome implied by the model after using the observed data to update our beliefs about the unknown parameters in the model. Simulating data from the posterior predictive distribution using the observed predictors is useful for checking the fit of the model. Drawing from the posterior predictive distribution at interesting values of the predictors also lets us visualize how a manipulation of a predictor affects (a function of) the outcome(s). With new observations of predictor variables we can use the posterior predictive distribution to generate predicted outcomes.

Usage

```
posterior_traj(object, m = 1, newdata = NULL, ids, limits = c(0.025,
  0.975), last_time = NULL, interpolate = TRUE, extrapolate = TRUE,
  interpolate_args = list(increments = 25), extrapolate_args = list(dist =
  NULL, prop = 0.5, increments = 25), re.form = NULL, fun = NULL,
  draws = NULL, seed = NULL, offset = NULL, ...)
```

Arguments

`object`	A fitted model object returned by the stan_jm modelling function. See stanjm-object.
`m`	Integer specifying the number of the longitudinal submodel
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the model matrix is used. If `newdata` is provided and any variables were transformed (e.g. rescaled) in the data used to fit the model, then these variables must also be transformed in `newdata`. This only applies if variables were transformed before passing the data to one of the modeling functions and *not* if transformations were specified inside the model formula. Also see the Note section below for a note about using the `newdata` argument with with binomial models.
`ids`	An optional vector providing a subset of IDs for whom the predicted curves should be plotted.
`limits`	A numeric vector of length 2 specifying the limits to use for both the credible and prediction intervals.

`last_time`	A scalar or
`interpolate`	A logical specifying whether to interpolate the estimated longitudinal trajectory in between the observation times. This can be used to achieve a smooth estimate of the longitudinal trajectory across the entire follow up time. If `TRUE` then the interpolation can be further controlled using the `interpolate_args` argument.
`extrapolate`	A logical specifying whether to extrapolate the estimated longitudinal trajectory beyond the time of the last longitudinal measurement. If `TRUE` then the extrapolation can be further controlled using the `extrapolate_args` argument.
`interpolate_args`	A named list with parameters controlling interpolation of the estimated longitudinal trajectory when `interpolate = TRUE`. The list can contain the following named elements: `increments`, a positive integer specifying the number of increments in time at which to calculate the estimated longitudinal response. See the **Examples** below.
`extrapolate_args`	A named list with parameters controlling extrapolation of the estimated longitudinal trajectory when `extrapolate = TRUE`. The list can contain the following named elements: `dist`, a positive scalar specifying the amount of time across which to extrapolate the longitudinal trajectory; `prop`, a positive scalar between 0 and 1 specifying the amount of time across which to extrapolate the longitudinal trajectory but represented as a proportion of the total observed follow up time for each individual; `increments`, a positive integer specifying the number of increments in time at which to calculate the estimated longitudinal response. See the **Examples** below.
`re.form`	If `object` contains `group-level` parameters, a formula indicating which group-level parameters to condition on when making predictions. `re.form` is specified in the same form as for `predict.merMod`. The default, `NULL`, indicates that all estimated group-level parameters are conditioned on. To refrain from conditioning on any group-level parameters, specify `NA` or `~0`. The `newdata` argument may include new *levels* of the grouping factors that were specified when the model was estimated, in which case the resulting posterior predictions marginalize over the relevant variables.
`fun`	An optional function to apply to the results. `fun` is found by a call to `match.fun` and so can be specified as a function object, a string naming a function, etc.
`draws`	An integer indicating the number of draws to return. The default and maximum number of draws is the size of the posterior sample.
`seed`	An optional `seed` to use.
`offset`	A vector of offsets. Only required if `newdata` is specified and an `offset` argument was specified when fitting the model.
`...`	Currently unused.

Value

A `draws` by `nrow(newdata)` matrix of simulations from the posterior predictive distribution. Each row of the matrix is a vector of predictions generated using a single draw of the model parameters from the posterior distribution.

Note

For binomial models with a number of trials greater than one (i.e., not Bernoulli models), if `newdata` is specified then it must include all variables needed for computing the number of binomial trials to use for the predictions. For example if the left-hand side of the model formula is `cbind(successes, failures)` then both `successes` and `failures` must be in `newdata`. The particular values of `successes` and `failures` in `newdata` do not matter so long as their sum is the desired number of trials. If the left-hand side of the model formula were `cbind(successes, trials - successes)` then both `trials` and `successes` would need to be in `newdata`, probably with `successes` set to `0` and `trials` specifying the number of trials. See the Examples section below and the *How to Use the rstanarm Package* for examples.

See Also

`pp_check` for graphical posterior predictive checks of the longitudinal submodel(s).

`pp_check` *Graphical posterior predictive checks for the longitudinal submodel*

Description

Various plots comparing the observed outcome variable from one of the longitudinal submodels y to simulated datasets y^{rep} from the posterior predictive distribution. This function is modelled on the `pp_check` function from the **rstanarm** package.

Usage

```
pp_check(object, m = 1, check = "distributions", nreps = NULL,
  seed = NULL, overlay = TRUE, test = "mean", ...)
```

Arguments

`object`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`m`	Integer specifying the number of the longitudinal submodel
`check`	The type of plot (possibly abbreviated) to show. One of `"distributions"`, `"residuals"`, `"scatter"`, `"test"`. See Details for descriptions.
`nreps`	The number of y^{rep} datasets to generate from the posterior predictive distribution (`posterior_predict`) and show in the plots. The default is `nreps=3` for `check="residuals"` and `nreps=8` for `check="distributions"`. If `check="test"`, `nreps` is ignored and the number of simulated datasets is the number of post-warmup draws from the posterior distribution. If `check="scatter"`, `nreps` is not ignored but defaults to the number of post-warmup draws.
`seed`	An optional `seed` to pass to `posterior_predict`.
`overlay`	For `check="distributions"` only, should distributions be plotted as density estimates overlaid in a single plot (`TRUE`, the default) or as separate histograms (`FALSE`)?
`test`	For `check="test"` only, a character vector (of length 1 or 2) naming a single function or a pair of functions. The function(s) should take a vector input and return a scalar test statistic. See Details and Examples.
`...`	Optional arguments to geoms to control features of the plots (e.g. `binwidth` if the plot is a histogram).

Details

Note that the posterior predictive checks here condition on all observed data, that is for a predicted observation $y_i^{rep}(t)$ the predictive distribution contains information from data that may have been observed later in time, that is some time s where $s > t$. The predictions are therefore suitable for assessing model fit, but should not be used to dynamic predictions where we want to condition only on previous measurements for the individual.

Descriptions of the plots corresponding to the different values of `check`:

`distributions` The distributions of y and `nreps` simulated y^{rep} datasets.

`residuals` The distributions of residuals computed from y and each of `nreps` simulated datasets. For binomial data, binned residual plots are generated (similar to `binnedplot`).

`scatter` If `nreps` is `NULL` then y is plotted against the average values of y^{rep}, i.e., the points $(y_n, \bar{y}_n^{rep})$, $n = 1, \dots, N$, where each y_n^{rep} is a vector of length equal to the number of posterior draws. If `nreps` is a (preferably small) integer, then only `nreps` y^{rep} datasets are simulated and they are each plotted separately against y.

`test` The distribution of a single test statistic $T(y^{rep})$ or a pair of test statistics over the `nreps` simulated datasets. If the `test` argument specifies only one function then the resulting plot is a histogram of $T(y^{rep})$ and the value of the test statistic in the observed data, $T(y)$, is shown in the plot as a vertical line. If two functions are specified then the plot is a scatterplot and $T(y)$ is shown as a large point.

Value

A ggplot object that can be further customized using the **ggplot2** package.

Note

For binomial data, plots of y and y^{rep} show the proportion of 'successes' rather than the raw count.

References

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis.* Chapman & Hall/CRC Press, London, third edition. (Ch. 6)

See Also

`posterior_predict` for drawing from the posterior predictive distribution for the longitudinal submodel, `posterior_survfit` for the estimated survival function based on the posterior distribution of the model parameters, and `ps_check` for graphical checks of the estimated survival function.

`print.stanjm` *Print method for stanjm objects*

Description

The `print` method for `stanjm` objects displays a compact summary of a joint model fitted using the `stan_jm` modelling function. For more detailed summary statistics and diagnostics use the `summary` method.

Usage

```
## S3 method for class 'stanjm'
print(x, digits = 3, ...)
```

Arguments

`x`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`digits`	Number of digits to use for formatting numbers.
`...`	Ignored.

Details

The output from the `print` method includes very brief summary statistics for the fixed effects parameter estimates (described below, and which are displayed separately for each longitudinal or event submodel), the additional parameters such as the residual error standard deviation, dispersion parameter, or the Weibull shape parameter or baseline hazard coefficients. Also displayed are the standard deviation and correlation parameter estimates for the random effects.

Point estimates: Point estimates are medians computed from the posterior sample generated via the MCMC simulations. The point estimates reported are the same as the values returned by the `coef` method.

Uncertainty estimates: The standard deviations (labeled MAD_SD in the print output) are computed from the same posterior sample used to calculate the point estimates and are proportional to the median absolute deviation (`mad`) from the median. Compared to the raw posterior standard deviation, the MAD_SD will be more robust for long-tailed distributions. These are the same as the values returned by `se`.

Value

Returns `x`, invisibly.

See Also

`summary.stanjm`, `stanjm-methods`

`priors` *Prior distributions and options for fitting Bayesian joint models via Stan*

Description

These functions are used to specify priors and related options when fitting a joint model using the `stan_jm` function, which is the main modeling function in the **rstanjm** package. The default priors are intended to be *weakly informative* in that they provide moderate regularlization and help stabilize computation. For many applications the defaults will perform well, but prudent use of more informative priors is encouraged. Uniform prior distributions are possible by setting the prior to `NULL` but, unless the data is very strong, they are not recommended and are *not* non-informative, since they give the same probability mass to implausible values as plausible ones.

Usage

```
normal(location = 0, scale = NULL)

student_t(df = 1, location = 0, scale = NULL)

cauchy(location = 0, scale = NULL)

hs(df = 3)

hs_plus(df1 = 3, df2 = 3)

decov(regularization = 1, concentration = 1, shape = 1, scale = 1)

priorLong_options(prior_scale_for_dispersion = 5, min_prior_scale = 1e-12,
  scaled = TRUE)

priorEvent_options(prior_scale_for_weibull = 5,
  prior_scale_for_splines = 10, min_prior_scale = 1e-12, scaled = TRUE)

priorAssoc_options(min_prior_scale = 1e-12)
```

Arguments

`location`	Prior location. For `normal` and `student_t` (provided that `df > 1`) this is the prior mean. For `cauchy` (which is equivalent to `student_t` with `df=1`), the mean does not exist and `location` is the prior median. The default value is 0.
`scale`	Prior scale. The default depends on the family (see Details).
`df, df1, df2`	Prior degrees of freedom. The default is 1 for `student_t`, in which case it is equivalent to `cauchy`. For the hierarchical shrinkage priors (`hs` and `hs_plus`) the degrees of freedom parameter(s) default to 3.
`regularization`	Exponent for an LKJ prior on the correlation matrix in the `decov` prior. The default is 1, implying a joint uniform prior.
`concentration`	Concentration parameter for a symmetric Dirichlet distribution. The defaults is 1, implying a joint uniform prior.
`shape`	Shape parameter for a gamma prior on the scale parameter in the `decov` prior. If `shape` and `scale` are both 1 (the default) then the gamma prior simplifies to the unit-exponential distribution.
`prior_scale_for_dispersion`	Prior scale for the standard error of the regression in Gaussian models, which is given a half-Cauchy prior truncated at zero.
`min_prior_scale`	Minimum prior scale for the intercept and coefficients.
`scaled`	A logical scalar, defaulting to `TRUE`. If `TRUE` the `prior_scale` is further scaled by the range of the predictor if the predictor has exactly two unique values and scaled by twice the standard deviation of the predictor if it has more than two unique values.
`prior_scale_for_weibull`	Prior scale for the shape parameter of the Weibull distribution for the baseline hazard in `stan_jm` (when the `base_haz` argument is set to "weibull"), which is given a half-Cauchy truncated at zero.

`prior_scale_for_splines`
: Prior scale for the coefficients of the restricted cubic splines for the baseline hazard in `stan_jm` (when the `base_haz` argument is set to "splines"), which are given normal distributions with location 0.

Details

The details depend on the family of the prior being used:

Student t family: Family members:

- `normal(location, scale)`
- `student_t(df, location, scale)`
- `cauchy(location, scale)`

For the prior distribution for the intercept, `location`, `scale`, and `df` should be scalars. For the prior for the other coefficients they can either be vectors of length equal to the number of coefficients (not including the intercept), or they can be scalars, in which case they will be recycled to the appropriate length. As the degrees of freedom approaches infinity, the Student t distribution approaches the normal distribution and if the degrees of freedom are one, then the Student t distribution is the Cauchy distribution.

If `scale` is not specified it will default to 10 for the intercept and 2.5 for the other coefficients, unless the probit link function is used, in which case these defaults are scaled by a factor of `dnorm(0)/dlogis(0)`, which is roughly 1.6.

Hierarchical shrinkage family: Family members:

- `hs(df)`
- `hs_plus(df1, df2)`

The hierarchical shrinkage priors are normal with a mean of zero and a standard deviation that is also a random variable. The traditional hierarchical shrinkage prior utilizes a standard deviation that is distributed half Cauchy with a median of zero and a scale parameter that is also half Cauchy. This is called the "horseshoe prior". The hierarchical shrinkage (`hs`) prior in the **rstanjm** package instead utilizes a half Student t distribution for the standard deviation (with 3 degrees of freedom by default), scaled by a half Cauchy parameter, as described by Piironen and Vehtari (2015). It is possible to change the `df` argument, the prior degrees of freedom, to obtain less or more shrinkage.

The hierarhical shrinkpage plus (`hs_plus`) prior is a normal with a mean of zero and a standard deviation that is distributed as the product of two independent half Student t parameters (both with 3 degrees of freedom (`df1`, `df2`) by default) that are each scaled by the same square root of a half Cauchy parameter.

These hierarchical shrinkage priors have very tall modes and very fat tails. Consequently, they tend to produce posterior distributions that are very concentrated near zero, unless the predictor has a strong influence on the outcome, in which case the prior has little influence. Hierarchical shrinkage priors often require you to increase the `adapt_delta` tuning parameter in order to diminish the number of divergent transitions. For more details on tuning parameters and divergent transitions see the Troubleshooting section of the *How to Use the rstanarm Package* vignette.

Covariance matrices: Family members:

- `decov(regularization, concentration, shape, scale)`

Covariance matrices are decomposed into correlation matrices and variances. The variances are in turn decomposed into the product of a simplex vector and the trace of the matrix. Finally, the trace is the product of the order of the matrix and the square of a scale parameter. This prior on a covariance matrix is represented by the `decov` function.

The prior for a correlation matrix is called LKJ whose density is proportional to the determinant of the correlation matrix raised to the power of a positive regularization parameter minus one. If `regularization = 1` (the default), then this prior is jointly uniform over all correlation matrices of that size. If `regularization > 1`, then the identity matrix is the mode and in the unlikely case that `regularization < 1`, the identity matrix is the trough.

The trace of a covariance matrix is equal to the sum of the variances. We set the trace equal to the product of the order of the covariance matrix and the *square* of a positive scale parameter. The particular variances are set equal to the product of a simplex vector — which is non-negative and sums to 1 — and the scalar trace. In other words, each element of the simplex vector represents the proportion of the trace attributable to the corresponding variable.

A symmetric Dirichlet prior is used for the simplex vector, which has a single (positive) `concentration` parameter, which defaults to 1 and implies that the prior is jointly uniform over the space of simplex vectors of that size. If `concentration > 1`, then the prior mode corresponds to all variables having the same (proportion of total) variance, which can be used to ensure the the posterior variances are not zero. As the `concentration` parameter approaches infinity, this mode becomes more pronounced. In the unlikely case that `concentration < 1`, the variances are more polarized.

If all the variables were multiplied by a number, the trace of their covariance matrix would increase by that number squared. Thus, it is reasonable to use a scale-invariant prior distribution for the positive scale parameter, and in this case we utilize a Gamma distribution, whose `shape` and `scale` are both 1 by default, implying a unit-exponential distribution. Set the `shape` hyperparameter to some value greater than 1 to ensure that the posterior trace is not zero.

If `regularization`, `concentration`, `shape` and / or `scale` are positive scalars, then they are recycled to the appropriate length. Otherwise, each can be a positive vector of the appropriate length, but the appropriate length depends on the number of covariance matrices in the model and their sizes. A one-by-one covariance matrix is just a variance and thus does not have `regularization` or `concentration` parameters, but does have `shape` and `scale` parameters for the prior standard deviation of that variable.

Value

A named list to be used internally by the model fitting function `stan_jm`.

---

ps_check *Graphical checks of the estimated survival function*

Description

This function plots the estimated marginal survival function based on draws from the posterior predictive distribution of the fitted joint model, and then overlays the Kaplan-Meier curve based on the observed data.

Usage

```
ps_check(object, check = "survival", limits = c("ci", "none"),
  draws = NULL, seed = NULL, xlab = NULL, ylab = NULL,
  ci_geom_args = NULL, ...)
```

Arguments

`object`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`check`	The type of plot to show. Currently only "survival" is allowed, which compares the estimated marginal survival function under the joint model to the estimated Kaplan-Meier curve based on the observed data.
`limits`	A quoted character string specifying the type of limits to include in the plot. Can be one of: `"ci"` for the Bayesian posterior uncertainty interval (often known as a credible interval); or `"none"` for no interval limits.
`draws`	An integer indicating the number of MCMC draws to use to to estimate the survival function. The default and maximum number of draws is the size of the posterior sample.
`seed`	An optional `seed` to use.
`xlab, ylab`	An optional axis label passed to `labs`.
`ci_geom_args`	Optional arguments passed to `geom_ribbon` and used to control features of the plotted interval limits.
`...`	Optional arguments passed to `geom_line` and used to control features of the plotted trajectory.

Value

A ggplot object that can be further customized using the **ggplot2** package.

See Also

`posterior_survfit` for the estimated marginal or subject-specific survival function based on draws of the model parameters from the posterior distribution, `posterior_predict` for drawing from the posterior predictive distribution for the longitudinal submodel, and `pp_check` for graphical checks of the longitudinal submodel.

Examples

```
if (!exists("examplejm")) example(examplejm)
# Compare estimated survival function to Kaplan-Meier curve
(ps <- ps_check(examplejm))
ps +
 ggplot2::scale_color_manual(values = c("red", "black")) + # change colors
 ggplot2::scale_size_manual(values = c(0.5, 3)) + # change line sizes
 ggplot2::scale_fill_manual(values = c(NA, NA)) # remove fill
```

`rstanarm-plots` *Plots for rstanarm models*

Description

Models fit using `algorithm='sampling'`, `"meanfield"`, or `"fullrank"` are compatible with a variety of plotting functions from the **rstan** package. Each function returns at least one `ggplot` object that can be customized further using the **ggplot2** package. The plotting functions described here can be called using the `plot method` for stanreg objects without loading the **rstan** package. For example usage see `plot.stanreg`.

Plotting functions

Posterior intervals and point estimates `stan_plot`

Traceplots `stan_trace`

Histograms `stan_hist`

Kernel density estimates `stan_dens`

Scatterplots `stan_scat`

Diagnostics for Hamiltonian Monte Carlo and the No-U-Turn Sampler `stan_diag`

Rhat `stan_rhat`

Ratio of effective sample size to total posterior sample size `stan_ess`

Ratio of Monte Carlo standard error to posterior standard deviation `stan_mcse`

Autocorrelation `stan_ac`

See Also

`plot.stanreg` for how to call the `plot` method, `shinystan` for interactive model exploration, `pp_check` for graphical posterior predicive checking.

Examples

```
# See examples at help("plot.stanreg", package = "rstanarm")
```

`shinystan` *Using the ShinyStan GUI with rstanjm models*

Description

The ShinyStan interface provides visual and numerical summaries of model parameters and convergence diagnostics.

Details

The `launch_shinystan` function will accept a `stanjm` object as input.

Faster launch times

For some **rstanjm** models ShinyStan may take a very long time to launch. If this is the case with one of your models you may be able to speed up `launch_shinystan` in one of several ways:

Prevent ShinyStan from preparing graphical posterior predictive checks: When used with a `stanjm` object (**rstanjm** model object) ShinyStan will draw from the posterior predictive distribution and prepare graphical posterior predictive checks before launching. That way when you go to the PPcheck page the plots are immediately available. This can be time consuming for models fit to very large datasets and you can prevent this behavior by creating a shinystan object before calling `launch_shinystan`. To do this use `as.shinystan` with optional argument ppd set to `FALSE` (see the Examples section below). When you then launch ShinyStan and go to the PPcheck page the plots will no longer be automatically generated and you will be presented with the standard interface requiring you to first specify the appropriate y and $yrep$, which can be done for many but not all **rstanarm** models.

Use a shinystan object: Even if you don't want to prevent ShinyStan from preparing graphical posterior predictive checks, first creating a shinystan object using `as.shinystan` can reduce *future* launch times. That is, `launch_shinystan(sso)` will be faster than `launch_shinystan(fit)`, where `sso` is a shinystan object and `fit` is a stanreg object. It still may take some time for `as.shinystan` to create `sso` initially, but each time you subsequently call `launch_shinystan(sso)` it will reuse `sso` instead of internally creating a shinystan object every time. See the Examples section below.

Examples

```
## Not run:
if (!exists("example_model")) example(example_model)

# Launch the ShinyStan app without saving the resulting shinystan object
if (interactive()) launch_shinystan(example_model)

# Launch the ShinyStan app (saving resulting shinystan object as sso)
if (interactive()) sso <- launch_shinystan(example_model)

# First create shinystan object then call launch_shinystan
sso <- shinystan::as.shinystan(example_model)
if (interactive()) launch_shinystan(sso)

# Prevent ShinyStan from preparing graphical posterior predictive checks that
# can be time consuming. example_model is small enough that it won't matter
# much here but in general this can help speed up launch_shinystan
sso <- shinystan::as.shinystan(example_model, ppd = FALSE)
if (interactive()) launch_shinystan(sso)

## End(Not run)
```

`stanjm-methods` *Methods for stanjm objects*

Description

S3 methods for stanjm objects. There are also several methods (listed in **See Also**, below) with their own individual help pages. A number of these methods will return a named list, with each element of the list containing the described item (for example, coefficients, standard errors, residuals, etc) for one of the longitudinal submodels or the event submodel.

Usage

```
## S3 method for class 'stanjm'
coef(object, ...)

## S3 method for class 'stanjm'
fitted(object, ...)

## S3 method for class 'stanjm'
log_lik(object, newdataLong = NULL, newdataEvent = NULL,
  ...)

## S3 method for class 'stanjm'
residuals(object, ...)

## S3 method for class 'stanjm'
se(object, ...)

## S3 method for class 'stanjm'
formula(x, fixed.only = FALSE, random.only = FALSE, ...)

## S3 method for class 'stanjm'
update(object, formulaLong., formulaEvent., ...,
  evaluate = TRUE)

## S3 method for class 'stanjm'
vcov(object, correlation = FALSE, ...)

## S3 method for class 'stanjm'
fixef(object, remove_stub = TRUE, ...)

## S3 method for class 'stanjm'
ngrps(object, ...)

## S3 method for class 'stanjm'
ranef(object, ...)

## S3 method for class 'stanjm'
sigma(object, ...)

## S3 method for class 'stanjm'
VarCorr(x, sigma = 1, ...)
```

Arguments

`object, x`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`...`	Ignored, except by the `update` method. See `update`.
`newdataLong, newdataEvent`	For `log_lik`, an optional data frame of new data (e.g. holdout data) to use when evaluating the log-likelihood. See the description of `newdataLong` and `newdataEvent` for `posterior_survfit`.
`fixed.only`	A logical specifying whether to only retain the fixed effect part of the longitudinal submodel formulas
`random.only`	A logical specifying whether to only retain the random effect part of the longitudinal submodel formulas
`formulaLong., formulaEvent.`	An updated formula for the longitudinal or event submodel. For a multivariate joint model `formulaLong.` should be a list of formulas, as described for the `formulaLong` argument in `stan_jm`.
`evaluate`	See `update`.
`correlation`	For `vcov`, if `FALSE` (the default) the covariance matrix is returned. If `TRUE`, the correlation matrix is returned instead.

`remove_stub`	Logical specifying whether to remove the string identifying the longitudinal or event submodel from each of the coefficient names.
`sigma`	Ignored (included for compatibility with `VarCorr`).

Details

Most of these methods are similar to the methods defined for objects of class 'lm', 'glm', 'glmer', etc. However there are a few exceptions:

`coef` Medians are used for point estimates. See the *Point estimates* section in `print.stanjm` for more details. `coef` returns a list equal to the length of the number of submodels. The first elements of the list are the coefficients from each of the fitted longitudinal submodels and are the same layout as those returned by `coef` method of the **lme4** package, that is, the sum of the random and fixed effects coefficients for each explanatory variable for each level of each grouping factor. The final element of the returned list is a vector of fixed effect coefficients from the event submodel.

`se` The `se` function returns standard errors based on `mad`. See the *Uncertainty estimates* section in `print.stanjm` for more details.

`confint` Not supplied, since the `posterior_interval` function should be used instead to compute Bayesian uncertainty intervals.

`residuals` Residuals are *always* of type `"response"` (not `"deviance"` residuals or any other type).

`log_lik` Returns the S by N pointwise log-likelihood matrix, where S is the size of the posterior sample and N is the number of individuals in the fitted model. The likelihood for a single individual is therefore the sum of the likelihood contributions from their observed longitudinal measurements and their event time data. Note: we use `log_lik` rather than defining a `logLik` method because (in addition to the conceptual difference) the documentation for `logLik` states that the return value will be a single number, whereas we return a matrix.

See Also

Other S3 methods for stanjm objects, which have separate documentation, including `as.matrix.stanjm`, `print.stanjm`, and `summary.stanjm`.

Also `posterior_interval` for an alternative to `confint`, and `posterior_predict`, `posterior_traj` and `posterior_survfit` for predictions based on the fitted joint model.

`stanjm-object` *Fitted model object in the* **rstanjm** *package*

Description

The main model fitting function in the **rstanjm** package (see `stan_jm`) returns an object of class `stanjm`, which is a list containing the components described below.

Value

The following components must be included in a `stanjm` object. In most cases the components return a named list, with each element of the list containing the described item (for example, coefficients, ses, residuals, etc) for one of the longitudinal submodels or the event submodel.

`coefficients` Point estimates for the longitudinal and event submodels. As described in `print.stanjm`.

`ses` Standard errors for the longitudinal and event submodels. Based on `mad`, as described in `print.stanjm`.

`residuals` Residuals for the longitudinal submodel(s) only. Of type `'response'`.

`fitted.values` Fitted mean values for the longitudinal submodel(s) only, based on the estimated parameters in `coefficients`. The fitted mean values are based on both the fixed and random effects. For models using a non-identity link function the linear predictors are transformed by the inverse link.

`linear.predictors` Linear fit on the link scale, for the longitudinal submodel(s) only. For linear models this is the same as `fitted.values`.

`covmat` Variance-covariance matrix for the coefficients (both fixed and random), evaluated separately for each of the longitudinal and event submodels. Calculation of the variance-covariance matrix is based on draws from the posterior distribution.

`n_markers` The number of longitudinal markers (submodels).

`n_subjects` The number of individuals in the fitted model.

`n_grps` The number of levels for each grouping factor (will be equal to `n_subjects` if the individual-level is the only clustering level).

`n_yobs` The number of observations for each longitudinal submodel.

`n_events` The number of non-censored event times.

`id_var,time_var` The names of the variables distinguishing between individuals, and representing time, in the longitudinal submodel.

`cnms` A list of column names of the random effects according to the grouping factors, collapsed across all longitudinal submodels. See `mkReTrms`. These can be obtained separately for each longitudinal submodel through the components `object$glmod[[1]]@cnms`, `object$glmod[[2]]@cnms`, etc.

`x` The design matrix (both fixed and random parts) for each of the longitudinal submodels, evaluated at the observation times.

`xq` The design matrices for each of the longitudinal and event submodels evaluated at the event/censoring times and the quadrature points. The first `n_subjects` rows correpond to the event/censoring times, the second `n_subjects` rows correspond to the first quadrature point, the third `n_subjects` rows correspond to the second quadrature point, and so on. The design matrices for the longitudinal submodels include both the fixed and random parts.

`y` The response for the longitudinal submodel(s).

`fr` The model frame (see `model.frame.merMod` or `model.frame.coxph`) for each of the longitudinal and event submodels, evaluated at the observation times in the original data (not evaluated at the quadrature points necessary for fitting the joint model). This component also includes the addition of the `id_var` and `time_var` variables in the model frame, even if they weren't present in the original model formula.

`eventtime,status` The event (or censoring) times and the status indicator for each individual.

`dataLong,dataEvent` The `dataLong` and `dataEvent` arguments.

`call` The matched call.

`formula` The model `formula` for each of the longitudinal and event submodels.

`family` The `family` object used for the longitudinal submodel(s).

`base_haz` A list containing information about the baseline hazard.

`assoc` A list containing information about the association structure for the joint model.

quadnodes The number of Gauss-Kronrod quadrature nodes used to evaluate the cumulative hazard in the joint likelihood function.

quadpoints A list containing the quadrature points for each individual, based on the event (or censoring) times and the number of quadnodes.

prior.info A list with information about the prior distributions used.

algorithm The estimation method used. Will be "sampling" for models estimated using stan_jm.

stanfit,stan_summary The object of stanfit-class returned by RStan and a matrix of various summary statistics from the stanfit object.

glmod The fitted model object(s) returned by a call(s) to lmer or glmer when estimating the separate longitudinal model(s) prior to fitting the joint model.

coxmod The fitted model object returned by a call to coxph when estimating the separate time-to-event model prior to fitting the joint model. This time-to-event model is evaluated using the observed data in dataEvent; it is not evaluated at the quadrature points necessary for fitting the joint model.

See Also

Objects of this class have a number of generic methods described in stanjm-methods, as well as print.stanjm, summary.stanjm, as.matrix.stanjm, as well as the non-generic functions posterior_traj, posterior_survfit, posterior_predict, posterior_interval, pp_check and ps_check.

---

stan_jm *Bayesian joint longitudinal and time-to-event models via Stan*

Description

Fits a shared parameter joint model for longitudinal and time-to-event (e.g. survival) data under a Bayesian framework using Stan.

Usage

```
stan_jm(formulaLong, dataLong, formulaEvent, dataEvent, time_var, id_var,
  family = gaussian, assoc = "etavalue", base_haz = c("weibull",
  "splines"), df, knots, quadnodes = 15, subsetLong, subsetEvent,
  na.action = getOption("na.action", "na.omit"), weights, offset, contrasts,
  centreLong = FALSE, centreEvent = FALSE, init = "model_based", ...,
  priorLong = normal(), priorLong_intercept = normal(),
  priorLong_ops = priorLong_options(), priorEvent = normal(),
  priorEvent_intercept = normal(), priorEvent_ops = priorEvent_options(),
  priorAssoc = normal(), priorAssoc_ops = priorAssoc_options(),
  prior_covariance = decov(), prior_PD = FALSE, adapt_delta = 0.75,
  max_treedepth = 9L, QR = FALSE, algorithm = c("sampling", "meanfield",
  "fullrank"))
```

Arguments

`formulaLong`	A two-sided linear formula object describing both the fixed-effects and random-effects parts of the longitudinal submodel (see `glmer` for details). For a multivariate joint model (i.e. more than one longitudinal marker) this should be a list of such formula objects, with each element of the list providing the formula for one of the longitudinal submodels.
`dataLong`	A data frame containing the variables specified in `formulaLong`. If fitting a multivariate joint model, then this can be either a single data frame which contains the data/variables for all the longitudinal submodels, or it can be a list of data frames where each element of the list provides the data for one of the longitudinal submodels.
`formulaEvent`	A two-sided formula object describing the event submodel. The left hand side of the formula should be a `Surv()` object. See `Surv`.
`dataEvent`	A data frame containing the variables specified in `formulaEvent`.
`time_var`	A character string specifying the name of the variable in `dataLong` which represents time.
`id_var`	A character string specifying the name of the variable in `dataLong` which distinguishes between individuals. This can be left unspecified if there is only one grouping factor (which is assumed to be the individual). If there is more than one grouping factor (i.e. clustering beyond the level of the individual) then the `id_var` argument must be specified.
`family`	The family (and possibly also the link function) for the longitudinal submodel(s). See `glmer` for details. If fitting a multivariate joint model, then this can optionally be a list of families, in which case each element of the list specifies the family for one of the longitudinal submodels.
`assoc`	A character string or character vector specifying the joint model association structure. Possible association structures that can be used include: "etavalue" (the default); "etaslope"; "muvalue"; "shared_b"; "shared_coef"; or "null". These are described in the **Details** section below. For a multivariate joint model, different association structures can optionally be used for each longitudinal submodel by specifying a list of character vectors, with each element of the list specifying the desired association structure for one of the longitudinal submodels. Specifying `assoc = NULL` will fit a joint model with no association structure (equivalent to fitting separate longitudinal and time-to-event models). See the **Examples** section below.
`base_haz`	A character string indicating which baseline hazard to use for the event submodel. Options are a Weibull baseline hazard (`"weibull"`, the default) or an approximated baseline hazard using B-splines (`"splines"`).
`df`	An optional positive integer specifying the degrees of freedom for the cubic splines if `base_haz = "splines"`. The default is 3.
`knots`	An optional numeric vector specifying the internal knot locations for the cubic splines if `base_haz = "splines"`. Cannot be specified if `df` is specified.
`quadnodes`	The number of nodes to use for the Gauss-Kronrod quadrature that is used to evaluate the cumulative hazard in the likelihood function. Options are 15 (the default), 11 or 7.
`subsetLong, subsetEvent`	Same as subset in `glm`. However, if fitting a multivariate joint model and a list of data frames is provided in `dataLong` then a corresponding list of subsets must be provided in `subsetLong`.

`na.action, contrasts`	Same as `glm`, but rarely specified.
`weights`	Experimental and should be used with caution. The user can optionally supply a 2-column data frame containing a set of 'prior weights' to be used in the estimation process. The data frame should contain two columns: the first containing the IDs for each individual, and the second containing the corresponding weights. The data frame should only have one row for each individual; that is, weights should be constant within individuals.
`offset`	Not yet implemented. Same as `glm`.
`centreLong, centreEvent`	A logical specifying whether the predictor matrix for the longitudinal submodel(s) or event submodel should be centred.
`init`	The method for generating the initial values for the MCMC. The default is `"model_based"`, which uses those obtained from fitting separate longitudinal and time-to-event models prior to fitting the joint model. Parameters that cannot be obtained from fitting separate longitudinal and time-to-event models are initialised at 0. This provides reasonable initial values which should aid the MCMC sampler. However, it is recommended that any final analysis should be performed with several MCMC chains each initiated from a different set of initial values; this can be obtained by setting `init = "random"`. Other possibilities for specifying `init` are those described for `stan`.
`...`	Further arguments passed to `sampling` (e.g. `iter`, `chains`, `cores`, etc.).
`priorLong, priorEvent, priorAssoc`	The prior distributions for the regression coefficients in the longitudinal submodel(s), event submodel, and the association parameter(s). Can be a call to `normal`, `student_t`, `cauchy`, `hs` or `hs_plus`. See `priors` for details. To to omit a prior — i.e., to use a flat (improper) uniform prior — set equal to `NULL`.
`priorLong_intercept, priorEvent_intercept`	The prior distributions for the intercepts in the longitudinal submodel(s) and event submodel. Can be a call to `normal`, `student_t` or `cauchy`. See `priors` for details. To to omit a prior — i.e., to use a flat (improper) uniform prior — set equal to `NULL`. (**Note:** If `centreLong` or `centreEvent` is set to `TRUE` then the prior distribution for the intercept is set so it applies to the value when all predictors are centered).
`priorLong_ops, priorEvent_ops, priorAssoc_ops`	Additional options related to prior distributions for the longitudinal submodel(s), event submodel, and the association parameter(s). To omit a prior on the dispersion parameters in the longitudinal submodel(s) set `priorLong_ops` to `NULL`. To omit a prior on the Weibull shape parameter (if `base_haz = "weibull"`) or the spline coefficients (if `base_haz = "splines"`) set `priorEvent_ops` to `NULL`. Otherwise see `priors`.
`prior_covariance`	Cannot be `NULL`; see `decov` for more information about the default arguments.
`prior_PD`	A logical (defaulting to `FALSE`) indicating whether to draw from the prior predictive distribution instead of conditioning on the data.
`adapt_delta`	See `adapt_delta` for details.
`max_treedepth`	A positive integer specifying the maximum treedepth for the non-U-turn sampler. See the `control` argument in `stan`.
`QR`	QR decomposition of the predictor matrix. Not yet implemented.
`algorithm`	Character string (possibly abbreviated) indicating the estimation approach to use. Currently, only "sampling" (for MCMC) is allowed.

Details

The `stan_jm` function can be used to fit a joint model (also known as a shared parameter model) for longitudinal and time-to-event data under a Bayesian framework. The joint model may be univariate (with only one longitudinal submodel) or multivariate (with more than one longitudinal submodel). Multi-level clustered data are allowed (e.g. patients within clinics), provided that the individual (e.g. patient) is the lowest level of clustering. The underlying estimation is carried out using the Bayesian C++ package Stan (`http://mc-stan.org/`).

For the longitudinal submodel a generalised linear mixed model is assumed with any of the `family` choices allowed by `glmer`.

For the event submodel a parametric proportional hazards model is assumed. The baseline hazard can be estimated using either a Weibull distribution (`base_haz = "weibull"`) or approximated using restricted cubic splines (`base_haz = "splines"`). If cubic splines are used then the degrees of freedom for the splines, or the internal knot locations, can be optionally specified. If the degrees of freedom are specified (through the `df` argument) then the knot locations are automatically generated based on the distribution of observed event times (not including censoring times) – see `ns` for details on the default knot placement. Otherwise the internal knot locations can be specified directly through the `knots` argument. If neither `df` or `knots` is specified, then the default is to set `df` equal to 3. It is not possible to specify both `df` and `knots`.

The association structure for the joint model can be based on any of the following parameterisations: current value of the linear predictor in the longitudinal submodel (`"etavalue"`); first derivative (slope) of the linear predictor in the longitudinal submodel (`"etaslope"`); current expected value of the longitudinal submodel (`"muvalue"`); shared individual-level random effects (`"shared_b"`); shared individual-level random effects which also incorporate the corresponding fixed effect as well as any corresponding random effects for clustering levels higher than the individual) (`"shared_coef"`); or no association structure (equivalent to fitting separate longitudinal and event models) (`"null"` or `NULL`). More than one association structure can be specified, however, not all possible combinations are allowed. By default, `"shared_b"` and `"shared_coef"` contribute all random effects to the association structure; however, a subset of the random effects can be chosen by specifying their indices between parentheses as a suffix, for example, "shared_b(1)" or "shared_b(1:3)" or "shared_b(1,2,4)", and so on.

Time-varying covariates are allowed in both the longitudinal and event submodels. These should be specified in the data in the same way as they normally would when fitting a separate longitudinal model using `lmer` or a separate time-to-event model using `coxph`. These time-varying covariates should be exogenous in nature, otherwise they would perhaps be better specified as an additional outcome (i.e. by including them as an additional longitudinal outcome/submodel in the joint model).

Bayesian estimation of the joint model is performed via MCMC. The Bayesian model includes independent priors on the regression coefficients for both the longitudinal and event submodels, including the association parameter(s) (in much the same way as the regression parameters in `stan_glm`) and priors on the terms of a decomposition of the covariance matrices of the group-specific parameters (in the same way as `stan_glmer`). See `priors` for more information about the priors distributions that are available.

Gauss-Kronrod quadrature is used to numerically evaluate the integral over the cumulative hazard in the likelihood function for the joint model. The accuracy of the numerical approximation can be controlled using the number of quadrature nodes, specified through the `quadnodes` argument. Using a higher number of quadrature nodes will result in a more accurate approximation.

Value

A stanjm object is returned.

Author(s)

Sam Brilleman <<sam.brilleman@monash.edu>>.

Whilst acknowledging this package draws heavily on code from the **rstanarm** package, of which I am not an author.

References

Stan Development Team. (2015). *Stan Modeling Language Users Guide and Reference Manual.* http://mc-stan.org/documentation/

See Also

stanjm-object, stanjm-methods, print.stanjm, summary.stanjm, posterior_traj, posterior_survfit, posterior_predict, posterior_interval, pp_check, ps_check.

Also see http://mc-stan.org/ for more information on the Stan C++ package used for model fitting.

Examples

```
#####
# Univariate joint model, with association structure based on the
# current value of the linear predictor
f1 <- stan_jm(formulaLong = logBili ~ year + (1 | id),
              dataLong = pbcLong,
              formulaEvent = Surv(futimeYears, death) ~ sex + trt,
              dataEvent = pbcSurv,
              time_var = "year",
              chains = 1, iter = 1000, warmup = 500)
summary(f1)

#####
# Univariate joint model, with association structure based on the
# current value of the linear predictor and shared random intercept
f2 <- stan_jm(formulaLong = logBili ~ year + (1 | id),
              dataLong = pbcLong,
              formulaEvent = Surv(futimeYears, death) ~ sex + trt,
              dataEvent = pbcSurv,
              assoc = c("etavalue", "shared_b"),
              time_var = "year",
              chains = 1, iter = 1000, warmup = 500)
summary(f2)

######
# Multivariate joint model, with association structure based
# on the current value of the linear predictor in each longitudinal
# submodel and shared random intercept from the second longitudinal
# submodel only (which is the first random effect in that submodel
# and is therefore indexed the '(1)' suffix in the code below)
mv1 <- stan_jm(
        formulaLong = list(
          logBili ~ year + (1 | id),
          albumin ~ sex + year + (1 + year | id)),
        dataLong = pbcLong,
        formulaEvent = Surv(futimeYears, death) ~ sex + trt,
        dataEvent = pbcSurv,
        assoc = list("etavalue", c("etavalue", "shared_b(1)")),
        time_var = "year",
        chains = 1, iter = 1000, warmup = 500, refresh = 25)
summary(mv1)

# To include both the random intercept and random slope in the shared
# random effects association structure for the second longitudinal
# submodel, we could specify the following
update(mv1, assoc = list("etavalue", c("etavalue", "shared_b"))
# which would be equivalent to
update(mv1, assoc = list("etavalue", c("etavalue", "shared_b(1,2)"))
# or
update(mv1, assoc = list("etavalue", c("etavalue", "shared_b(1:2)"))

######
# Multivariate joint model, estimated using multiple MCMC chains
# run in parallel across all available PC cores
mv2 <- stan_jm(formulaLong = list(
        logBili ~ year + (1 | id),
        albumin ~ sex + year + (1 +  year | id)),
        dataLong = pbcLong,
        formulaEvent = Surv(futimeYears, death) ~ sex + trt,
        dataEvent = pbcSurv,
        assoc = list("etavalue", c("etavalue", "shared_b(1)")),
        time_var = "year",
        chains = 3, iter = 1000, warmup = 500, refresh = 25,
        cores = parallel::detectCores())
summary(mv2)
```

`summary.stanjm` *Summary method for stanjm objects*

Description

The summary method for `stanjm` objects returns a summary of parameter estimates and MCMC convergence diagnostics (Monte Carlo error, effective sample size, Rhat) for a joint model fitted using the `stan_jm` modelling function.

Usage

```
## S3 method for class 'stanjm'
summary(object, pars = c("long", "event"),
  regex_pars = NULL, probs = NULL, digits = 3, ...)

## S3 method for class 'summary.stanjm'
print(x, digits = max(1, attr(x, "print.digits")),
  ...)
```

```
## S3 method for class 'summary.stanjm'
as.data.frame(x, ...)
```

Arguments

`object`	A fitted model object returned by the `stan_jm` modelling function. See `stanjm-object`.
`pars`	An optional character vector specifying a subset of parameters to display. Parameters can be specified by name or several shortcuts can be used. Using `pars = "long"` will display the parameter estimates for the longitudinal submodels only (excluding random effects, but including dispersion parameters). Using `pars = "event"` will display the parameter estimates for the event submodel only, including any association parameters. Using `pars = "assoc"` will display only the association parameters. Using `pars = "fixef"` will display all fixed effects, but not the random effects or the additional parameters such as dispersion, etc. Using `pars = "beta"` will display only the fixed effect regression coefficients (excluding the intercept terms). Using `pars = "alpha"` will display only the fixed effect intercept terms. The estimates for the random effects can be selected using `pars = "b"` or `pars = "varying"`. If both `pars` and `regex_pars` are set to `NULL` then all parameters are selected, including the log posterior. See **Examples**.
`regex_pars`	An optional character vector of regular expressions to use for parameter selection. `regex_pars` can be used in place of `pars` or in addition to `pars`.
`probs`	An optional numeric vector of probabilities passed to `quantile`, for calculating the quantiles of the posterior distribution for each parameter.
`digits`	Number of digits to use for formatting numbers when printing. When calling `summary`, the value of digits is stored as the `"print.digits"` attribute of the returned object.
`...`	Currently ignored.
`x`	An object of class `"summary.stanjm"`.

Value

The `summary` method returns an object of class `"summary.stanjm"`, which is a matrix of summary statistics and diagnostics, with additional information stored as attributes. The `print` method for `summary.stanjm` objects is called for its side effect and just returns its input. The `as.data.frame` method for `summary.stanjm` objects converts the matrix to a data.frame, preserving row and column names but dropping the `print`-related attributes.

See Also

`print.stanjm`, `stanjm-methods`

Examples

```
if (!exists("examplejm")) example(examplejm)

# Only showing 10th and 90th percentile
summary(examplejm, probs = c(0.1, 0.9))

# These produce the same output for this example,
# but the first method can be used for any model
summary(examplejm, pars = c("long"))
summary(examplejm, pars = c("Long1|(Intercept)",
                             "Long1|year",
                             "Long1|sigma"))

# Only show parameters for event submodel
summary(examplejm, pars = "event")

# Only show the association parameter for the current
# value of the linear predictor from the longitudinal submodel
summary(examplejm, pars = "Assoc|Long1:eta-value")
# or since there is only one association parameter in the
# model we can just use the following shortcut
summary(examplejm, pars = "assoc")

# Only show random effects parameters
summary(examplejm, pars = "b")
as.data.frame(summary(example_model, pars = "b"))

# To obtain only the random intercepts we could also use a
# regular expression in the regex argument as follows
summary(examplejm, pars = NULL,
        regex_pars = "\\(Intercept\\)\\sid")
```

Index