Package 'mpr'

July 23, 2025

Type Package

Title Multi-Parameter Regression (MPR)

Version 1.0.6

Date 2021-12-23

Author Kevin Burke

Maintainer Kevin Burke <kevin.burke@ul.ie>

Depends R (>= 2.14)

Imports survival

Description Fitting Multi-Parameter Regression (MPR) models to right-censored survival data. These are flexible parametric regression models which extend standard models, for example, proportional hazards. See Burke & MacKenzie (2016) <doi:10.1111/biom.12625> and Burke et al (2020) <doi:10.1111/rssc.12398>.

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2022-01-08 01:42:44 UTC

Contents

addterm 2
Distributions 4
mpr 6
predict.mpr 9
stepmpr 10
summary.mpr 12
update.mpr 14

Index **17**

addterm	*All Possible Single-Term Additions to / Deletions from a Multi-Parameter Regression (MPR) Model*

Description

Identifies all models which arise via single-term additions to / deletions from a component (or, simultaneously, multiple components) of the supplied `mpr` model, fits all such models and summarises these models in a table.

Usage

```
addterm(object, upper = ~ ., comp = 1:(object$ncomp),
        aic = TRUE, bestmodel = object, ...)

dropterm(object, lower = ~ 1, comp = 1:(object$ncomp),
         aic = TRUE, bestmodel = object, ...)
```

Arguments

`object`	an object of class "mpr" which is the result of a call to `mpr`.
`upper`	a one-sided `formula` (used in `addterm`) specifying a maximal model which must include the current one.
`lower`	a one-sided `formula` (used in `dropterm`) specifying a minimal model which must be within the current one.
`comp`	a numeric value (or vector) indicating the regression component (or components) where (simultaneous) additions / deletions occur. Note that "1" $= \lambda$, "2" $= \gamma$ and "3" $= \rho$. For more information on the various components, see `mpr` and `distributions`.
`aic`	logical. If `TRUE`, AIC is used as the basis for determining the best model among those considered. If `FALSE`, BIC is used.
`bestmodel`	an initial best model which, by default, is the supplied current model. This argument is used within the `stepmpr` function but is unlikely to be used directly by the end user.
`...`	additional arguments to be passed to internal methods.

Details

The hierarchy is respected when considering terms to be added or dropped, e.g., all main effects contained in a second-order interaction must remain.

When using `addterm`, the terms in the `upper` model formula must be a *superset* of the terms for each regression component indicated by `comp`. For example, if component 1 is `~ a + b + c` and component 2 is `~ a + b` (and terms are to be added to both simultaneously, i.e., `comp=1:2`), then `upper = ~ a + b + c + d` is acceptable and means that the variable `d` will be added simultaneously to components 1 and 2 (this can be written more compactly as `upper = ~ . + d`). On the other hand, `~ a + b + d` is not acceptable since its terms do not form a superset of the terms in component 1 (however, this would be acceptable if we were only considering component 2, i.e., if `comp=2`).

When using `dropterm`, the terms in the `lower` model formula must be a *subset* of the terms for each regression component indicated by `comp`. Again, if component 1 is `~ a + b + c` and component 2 is `~ a + b` (and terms are to be dropped from both simultaneously, i.e., `comp=1:2`), then `lower = ~ a` is acceptable and means that the variable `b` will be dropped simultaneously from components 1 and 2 (this can be written more compactly as `lower = ~ . - b`). On the other hand, `~ c` is not acceptable since its terms do not form a subset of the terms in component 2 (however, this would be acceptable if we were only considering component 1, i.e., if `comp=1`).

To summarise the above two paragraphs, the `upper` formula must *contain* each formula corresponding to the components under consideration whereas the `lower` formula must *be contained within* each of these formulae.

Value

A `list` containing the following components:

`modeltab`	a table containing information about each of the fitted models. This information comes from the "`model`" element of each of the `mpr` objects - see `mpr` for details.
`bestmodel`	the model with the lowest AIC (or BIC if `aic = FALSE`) among the fitted models including the initial `bestmodel` passed to the `addterm` / `dropterm` function.

Author(s)

Kevin Burke.

See Also

`mpr`, `stepmpr`, `update.mpr`.

Examples

```
# Veterans' administration lung cancer data
veteran <- survival::veteran
head(veteran)

# null model
mod1 <- mpr(Surv(time, status) ~ list(~ 1, ~ 1), data=veteran)
mod1 # family = "Weibull" by default

# consider adding trt and celltype to component 1
addterm(mod1, ~ trt + celltype, comp=1)

# consider adding trt and celltype to components 1 and 2 simultaneously
addterm(mod1, ~ trt + celltype, comp=1:2)$modeltab

# further examples
mod2 <- mpr(Surv(time, status) ~ list(~ trt + celltype, ~ trt + karno),
            data=veteran)
dropterm(mod2, ~ 1, comp=1:2)$modeltab
dropterm(mod2, ~ 1, comp=1)$modeltab
dropterm(mod2, ~ 1, comp=2)$modeltab

# does nothing since celltype is only in component 1
dropterm(mod2, ~ . - celltype, comp=1:2)$modeltab

# removes celltype from component 1
dropterm(mod2, ~ . - celltype, comp=1)$modeltab
```

`Distributions` *Distributions in the* `mpr` *Package*

Description

Information on the distributions currently available within the `mpr` package.

Details

When fitting a Multi-Parameter Regression (MPR) model to data, the underlying distribution is selected using the "`family`" argument in the `mpr` function.

Currently the `mpr` package includes distributions which have upto three parameters:

1 λ. This is a *scale* parameter which controls the overall magnitude of the hazard function and is typically the "interest" parameter in standard Single-Parameter Regression (SPR) models. The Multi-Parameter Regression (MPR) framework is more general and considers all parameters to be of interest.

2 γ. This is a *shape* parameter which controls the time evolution of the hazard.

3 ρ. This is an additional *shape* parameter which controls the time evolution of the hazard (available within the Burr and PGW distributions).

The MPR framework allows these parameters to depend on covariates as follows:

$$g_1(\lambda) = x^T\beta$$

$$g_2(\gamma) = z^T\alpha$$

$$g_3(\rho) = w^T\tau$$

where $g_1(.)$, $g_2(.)$ and $g_3(.)$ are appropriate link functions (log-link for positive parameters and identity-link for unconstrained parameters), x, z and w are covariate vectors, which may or may not contain covariates in common, and β, α and τ are the corresponding vectors of regression coefficients.

The distributions currently available are described below in terms of their hazard functions:

`family`	**Hazard** $h(t)$	**Parameters**	**Note**
`Weibull`	$\lambda\gamma t^{\gamma-1}$	$\lambda > 0, \gamma > 0$	SPR(λ) = PH

`WeibullAFT`	$\lambda\gamma(\lambda t)^{\gamma-1}$	$\lambda > 0, \gamma > 0$	$\text{SPR}(\lambda) = \text{AFT}$
`Gompertz`	$\lambda \exp(\gamma t)$	$\lambda > 0, \gamma \in (-\infty, \infty)$	$\text{SPR}(\lambda) = \text{PH}$
`Loglogistic`	$\frac{\lambda\gamma t^{\gamma-1}}{1+\lambda t^{\gamma}}$	$\lambda > 0, \gamma > 0$	$\text{SPR}(\lambda) = \text{PO}$
`TDL`	$\frac{\exp(\gamma t+\lambda)}{1+\exp(\gamma t+\lambda)}$	$\lambda \in (-\infty, \infty), \gamma \in (-\infty, \infty)$	—
`Burr`	$\frac{\lambda\gamma t^{\gamma-1}}{1+\lambda\rho t^{\gamma}}$	$\lambda > 0, \gamma > 0, \rho > 0$	—
`PGW`	$\lambda\gamma\rho t^{\gamma-1}(1 + t^{\gamma})^{\rho-1}$	$\lambda > 0, \gamma > 0, \rho > 0$	$\text{SPR}(\lambda) = \text{PH}$

The acronymns which appear in the table above are:

SPR(λ) a Single-Parameter Regression (SPR) model where covariates enter through the scale parameter, λ. For example, in the row corresponding to the `Weibull` model, "$\text{SPR}(\lambda) = \text{PH}$" means that the Weibull $\text{SPR}(\lambda)$ model is a PH model. Thus, this standard parametric PH model is generalised via the Weibull MPR model.

PH proportional hazards.

AFT accelerated failure time.

PO proportional odds.

TDL time-dependent logistic.

PGW power generalised Weibull.

Author(s)

Kevin Burke.

See Also

`mpr`

Examples

```
# Veterans' administration lung cancer data
veteran <- survival::veteran
head(veteran)

# Weibull MPR treatment model
mpr(Surv(time, status) ~ list(~ trt, ~ trt), data=veteran, family="Weibull")

# Burr MPR treatment model
mpr(Surv(time, status) ~ list(~ trt, ~ trt, ~ trt), data=veteran,
    family="Burr")
```

mpr *Fitting a Multi-Parameter Regression (MPR) model.*

Description

Fits a Multi-Parameter Regression (MPR) model using a Newton-type algorithm via the `nlm` function.

Usage

```
mpr(formula, data, family = "Weibull", init, iterlim = 1000, ...)
```

Arguments

`formula`	a two-sided `formula` object with the response on the left hand side of the `~` operator and a `list` of one-sided `formula` objects on the right hand side (one for each regression component in the `mpr` model). The response must be a right-censored survival object as returned by the `Surv` function. See “Details” for more information on the struture of the formula within the `mpr` function as it differs from standard regression models.
`data`	an optional `data.frame` containing the variables in the model. If missing, the variables are taken from the `environment` from which `mpr` is called.
`family`	the name of the parametric distribution to be used in the model. See `distributions` for the list of distributions currently available.
`init`	an optional vector of initial values for the optimisation routine. If missing, default values are used. One may also set `init="random"` to randomly generate initial values.
`iterlim`	a positive integer specifying the maximum number of iterations to be performed before the optimisation procedure is terminated. This is supplied to `nlm`.
`...`	additional arguments to be passed to `nlm`.

Details

Multi-Parameter Regression (MPR) models are generated by allowing *multiple* distributional parameters to depend on covariates, for example, both the scale and shape parameters. This is in contrast to the more typical approach where covariates enter a model only through *one* distributional parameter. As these standard models have a single regression component, we may refer to them as Single Parameter Regression (SPR) models and, clearly, they are special cases of MPR models. The parameter through which covariates enter such SPR models may be referred to as the “interest” parameter since it generally has some specific subject-matter importance. However, this standard approach neglects other parameters which may also be important in describing the phenomenon at hand. The MPR approach generalises the standard SPR approach by viewing all distributional parameters as interest parameters in which covariate effects can be investigated.

In the context of survival analysis (currently the focus of the `mpr` package), the Weibull model is one of the most popular parametric models. Its hazard function is given by

$$h(t) = \lambda\gamma t^{\gamma-1}$$

where $\lambda > 0$, the scale parameter, controls the overall magnitude of $h(t)$ and $\gamma > 0$, the shape parameter, controls its time evolution. In the standard SPR Weibull model, λ depends on covariates via $\log \lambda = x^T \beta$ leading to a proportional hazards (PH) model. The MPR model generalises this by allowing *both* parameters to depend on covariates as follows

$$\log \lambda = x^T \beta$$

$$\log \gamma = z^T \alpha$$

where x and z are the scale and shape covariate vectors (which may or may not contain covariates in common) and β and α are the corresponding regression coefficients.

Note that the log-link is used above to ensure positivity of the parameters. More generally, we may have

$$g_1(\lambda) = x^T \beta$$

$$g_2(\gamma) = z^T \alpha$$

where $g_1(\cdot)$ and $g_2(\cdot)$ are appropriate link functions. The `mpr` function does not allow the user to alter these link functions but, rather, uses the following default link functions: **log-link** (for parameters which must be *positive*) and **identity-link** (for parameters which are *unconstrained*). Although the two-parameter Weibull distribution is discussed here (due to its popularity), other distributions may have additional shape parameters, for example,

$$g_3(\rho) = w^T \tau$$

where w and τ are the vectors of covariates and regression coefficients for this additional shape component. See `distributions` for further details on the distributions currently available.

The struture of the `formula` within the `mpr` function is, for example, `Surv(time, status) ~ list(~ x1 + x2, ~ x1)` which clearly generalises the typical `formula` used in standard models (i.e., those with only one regression component) in the sense that the right hand side is a `list` of one-sided `formula` objects. Note the requirement that the `~` operator precedes each element within the `list`. Specifically, the example shown here represents the case where the covariates `x1` and `x2` appear in the first regression component, λ, and the covariate `x2` appears in the second regression component, γ. If there was a third regression component, ρ, then there would be an additional component in the `list`, for example, `Surv(time, status) ~ list(~ x1 + x2, ~ x1, ~ x1)`. The `mpr` function also accepts more typical two-sided `formula` objects, such as `Surv(time, status) ~ x1 + x2`, which imply that the terms on the right hand side appear in *each* of the regression components.

Value

`mpr` returns an object of class "`mpr`".

The function `summary` (i.e., `summary.mpr`) can be used to obtain and print a summary of the results. The the generic accessor function `coefficients` extracts the `list` of regression coefficient vectors. One can also apply `predict` (i.e., `predict.mpr`) to predict various quantites from the fitted `mpr` model. A stepwise variable selection procedure has been implemented for `mpr` models - see `stepmpr`.

An object of class `mpr` is a `list` containing the following components:

`model`	a `data.frame` containing useful information about the fitted model with the following headings:

`family` the chosen `distribution`.

`npar` number of estimated parameters in the fitted model.

`loglike` value of the log-likelihood.

`aic` value of the AIC (Akaike Information Criterion).

`bic` value of the BIC (Bayesian Information Criterion).

`code` an integer indicating why the Newton optimisation procedure terminated (for more details on this stop-code see `nlm`) where, in particular, “1” means “relative gradient is close to zero”.

`coefficients`	a `list` whose elements are named vectors of coefficients (one vector per regression component).
`vcov`	the variance-covariance matrix for the estimates.
`gradient`	the values of the (negative) score functions from `nlm`.
`ncomp`	the number of regression components in the model, i.e., the number of distributional parameters in the underlying `distribution`.
`formula`	the `formula` supplied.
`xvars`	a record of the names of all variables (i.e., covariates) used in fitting.
`xlevels`	a record of the levels of any `factor`s (i.e., categorical variables) used in fitting.
`call`	the matched call.

Author(s)

Kevin Burke.

See Also

`distributions`, `summary.mpr`, `predict.mpr`, `stepmpr`.

Examples

```
# Veterans' administration lung cancer data
veteran <- survival::veteran
head(veteran)

# treatment variable, "trt", in scale (lambda) and shape (gamma)
# components of a Weibull model
mpr(Surv(time, status) ~ list(~ trt, ~ trt), data=veteran, family="Weibull")

# same as first model
mpr(Surv(time, status) ~ trt, data=veteran, family="Weibull")

# now with "celltype" also appearing in the scale
mpr(Surv(time, status) ~ list(~ trt + celltype, ~ trt), data=veteran,
    family="Weibull")

# trt in scale only (this is a PH Weibull model)
mpr(Surv(time, status) ~ list(~ trt, ~ 1), data=veteran, family="Weibull")
```

```
# trt in all three components (scale and two shape parameters) of a Burr model
mpr(Surv(time, status) ~ list(~ trt, ~ trt, ~ trt), data=veteran,
    family="Burr")

# use of summary
mod1 <- mpr(Surv(time, status) ~ list(~ trt, ~ trt), data=veteran)
summary(mod1)
```

`predict.mpr` *Predict method for Multi-Parameter Regression (MPR) Fits*

Description

Survival predictions based on `mpr` objects.

Usage

```
## S3 method for class 'mpr'
predict(object, newdata, type = c("survivor", "hazard", "percentile"),
        tvec, prob = 0.5, ...)
```

Arguments

`object`	an object of class "`mpr`" which is the result of a call to `mpr`.
`newdata`	`data.frame` in which to look for variables with which to predict.
`type`	type of prediction which may be a survivor function, hazard function or percentile value.
`tvec`	vector of times at which the predicted survivor or hazard function will be evaluated. Only required if `type` is `"survivor"` or `"hazard"`.
`prob`	numeric value between 0 and 1 (i.e., probability) indicating the percentile to be predicted. By default `prob = 0.5` which corresponds to the median value. Only required if `type` is `"percentile"`.
`...`	further arguments passed to or from other methods.

Value

A matrix of predictions whose rows correspond to the rows of `newdata`. When `type` is `"survivor"` or `"hazard"`, this matrix of predictions has columns corresponding to `tvec`. However, when `type` is `"percentile"`, the matrix only has one column.

Author(s)

Kevin Burke.

See Also

`mpr`, `summary.mpr`

Examples

```
library(survival)

# Veterans' administration lung cancer data
veteran <- survival::veteran
head(veteran)

# Weibull MPR treatment model
mod1 <- mpr(Surv(time, status) ~ list(~ trt, ~ trt), data=veteran,
            family="Weibull")

# predicted survivor function evaluated at four times
predict(mod1, newdata=data.frame(trt=c(1,2)), type="survivor",
        tvec=c(25, 50, 100, 150))

# predicted percentiles
predict(mod1, newdata=data.frame(trt=c(1,2)), type="percentile", prob=0.5)
predict(mod1, newdata=data.frame(trt=c(1,2)), type="percentile", prob=0.1)

# comparing predicted survivor functions to Kaplan-Meier curves
KM <- survfit(Surv(time, status) ~ trt, data=veteran)
plot(KM, col=1:2)
tvec <- seq(0, max(KM$time), length=100)
Stpred <- predict(mod1, newdata=data.frame(trt=c(1,2)), type="survivor",
                  tvec=tvec)
lines(tvec, Stpred[1,])
lines(tvec, Stpred[2,], col=2)
```

`stepmpr` *Stepwise Selection Procedure for Multi-Parameter Regression (MPR) Models*

Description

Applies a stepwise selection procedure to an object of class “`mpr`” to find the best model in the sense of AIC (or BIC).

Usage

```
stepmpr(object, scope = list(lower = ~ 1, upper = ~ .),
        comp = 1:(object$ncomp), direction = c("both", "backward", "forward"),
        joint = TRUE, jointonly = FALSE, aic = TRUE, trace = 3, ...)
```

Arguments

`object`	an object of class “`mpr`” which is the result of a call to `mpr`.
`scope`	either a single formula defining the `upper` (maximal) model or a list containing two formulae - the `lower` (minimal) and `upper` (maximal) models respectively. See “Details” for further information.

`comp`	a numeric vector indicating the regression component(s) to which the selection procedure should be applied. Note that “1” $= \lambda$, “2” $= \gamma$ and “3” $= \rho$. For more information on the various components, see `mpr` and `distributions`.
`direction`	the mode of stepwise search, which can be one of `"both"`, `"backward"`, or `"forward"`, with a default of `"both"`.
`joint`	logical. If `TRUE`, the selection procedure carries out *joint* component (i.e., simultaneous) steps in addition to *individual* component steps. If `FALSE`, only individual component steps are carried out. See “Details” for more information.
`jointonly`	logical. If `TRUE`, the selection procedure only carries out *joint* component steps.
`aic`	logical. If `TRUE`, AIC is used as the basis for determining the best model among those considered. If `FALSE`, BIC is used.
`trace`	if positive, information is printed during the running of `stepmpr`. Larger values may give more detailed information.
`...`	additional arguments to be passed to internal methods.

Details

The function `stepmpr` uses repeated calls to `addterm` and `dropterm` and is based on the idea that variable selection should be applied to each component individually *and* to all components jointly (when `joint = TRUE`). As an example, consider the case where forward selection (`direction = "forward"`) will be carried out in components 1 and 2 individually (`comp = 1:2`) *and* jointly (`joint = TRUE`). At a given iteration of the algorithm, the following single-term additions are then carried out:

individual step `1`: each term currently absent from component 1 will be considered.

individual step `2`: each term currently absent from component 2 will be considered.

joint step `1&2`: each term currently absent from both components 1 and 2 will be considered.

The reason for the joint step is to account for the possibility that a covariate may only appear significant when it is present simultaneously in *both* regression components. This situation can arise as the variance-covariance matrix for the estimated regression coefficients is typically not block diagonal with respect to the regression components and, in particular, coefficients for the same covariate in different components are typically highly correlated. Of course, the `stepmpr` function has the flexibility to carry individual steps only, joint steps only or individual steps in a particular component only as the end user prefers. See “Examples” below.

The set of models searched is determined by the `scope` argument which is either a single `upper` formula or a list whose elements are `lower` and `upper` formulae. The `upper` formula must *contain* each formula corresponding to the components under consideration (as indicated by `comp`) whereas the `lower` formula must *be contained within* each of these formulae. For more information on the use of `lower` and `upper`, see `addterm`.

If `scope` is missing, the `lower` model is simply the null model (i.e., a model with no covariates) and the `upper` model is formed using terms from the initial model; specifically, all terms from the regession components under consideration (as indicated by `comp`) are used in the `upper` formula.

Value

A final “best” `mpr` model as selected using the stepwise procedure.

Author(s)

Kevin Burke.

See Also

`mpr`, `dropterm`, `addterm`, `update.mpr`.

Examples

```
# Veterans' administration lung cancer data
veteran <- survival::veteran
head(veteran)

#######
mod0 <- mpr(Surv(time, status) ~ 1, data=veteran)
mod0 # family = "Weibull" by default

# the "upper" model formula (by default the lower will be ~ 1)
scope <- ~ trt + celltype

stepmpr(mod0, scope)
stepmpr(mod0, scope, direction="forward", aic=FALSE)

# individual steps only
stepmpr(mod0, scope, joint=FALSE)

# joint steps only
stepmpr(mod0, scope, jointonly=TRUE)

# component 1 only (and, hence, only individual steps)
stepmpr(mod0, scope, comp=1)

#######
mod1 <- mpr(Surv(time, status) ~ trt + celltype, data=veteran)
mod1

stepmpr(mod1)
stepmpr(mod1, scope = ~ .^2)

# "lower" model formula forces trt to stay in
stepmpr(mod1, scope = list(~trt, ~.))
```

`summary.mpr` *Summarising Multi-Parameter Regression (MPR) Fits*

Description

`summary` method for class "`mpr`"

Usage

```
## S3 method for class 'mpr'
summary(object, overall = TRUE, ...)
```

Arguments

`object`	an object of class “`mpr`” which is the result of a call to `mpr`.
`overall`	logical. If `TRUE`, p-values testing the overall effect of a covariate are shown. See “Details” for more information.
`...`	further arguments passed to or from other methods.

Details

The function `print.summary.lm` produces a typical table of coefficients, standard errors and p-values along with “significance stars”. In addition, a table of overall p-values are shown.

Multi-Parameter Regression (MPR) models are defined by allowing mutliple distributional parameters to depend on covariates. The regression components are:

$$g_1(\lambda) = x^T\beta$$

$$g_2(\gamma) = z^T\alpha$$

$$g_3(\rho) = w^T\tau$$

and the table of coefficients displayed by `print.summary.lm` follows this ordering. Furthermore, the names of the coefficients in the table are proceeded by “`.b`” for β coefficients, “`.a`” for α coefficients and “`.t`” for τ coefficients to avoid ambiguity.

Let us assume that a covariate c, say, appears in both the λ and γ regression components. The standard table of coefficients provides p-values corresponding to the following null hypotheses:

$$H_0 : \beta_c = 0$$

$$H_0 : \alpha_c = 0$$

where β_c and α_c are the regression coefficients of c (one for each of the two components in which c appears). However, in the context of MPR models, it may be of interest to test the hypothesis that the **overall** effect of c is zero, i.e., that its β *and* α effects are jointly zero:

$$H_0 : \beta_c = \alpha_c = 0$$

Thus, if `overall=TRUE`, `print.summary.lm` displays a table of such “overall p-values”.

Value

The function `summary.mpr` returns a `list` containing the following components:

`call`	the matched call from the `mpr` object.
`model`	a `data.frame` containing useful information about the fitted model. This is the same as the “`model`” element of the `mpr` object - see `mpr` for details.
`coefmat`	a typical coefficient matrix whose columns are the estimated regression coefficients, standard errors and p-values.
`overallpmat`	a matrix containing the overall p-values as described above in “Details”.

Author(s)

Kevin Burke.

See Also

`mpr`, `predict.mpr`.

Examples

```
# Veterans' administration lung cancer data
veteran <- survival::veteran
head(veteran)

# Weibull MPR treatment model (family = "Weibull" by default)
mod1 <- mpr(Surv(time, status) ~ list(~ trt, ~ trt), data=veteran)

summary(mod1)
```

`update.mpr` *Update and Re-fit a Multi-Parameter Regression (MPR) Model Call*

Description

Updates the right-hand side of the `formula` and re-fits the `mpr` model.

Usage

```
## S3 method for class 'mpr'
update(object, new, comp = 1:(object$ncomp), ...)
```

Arguments

`object`	an object of class "`mpr`" which is the result of a call to `mpr`.
`new`	either a one-sided `formula` (in which case the `comp` argument is also required) or a `list` of one-sided `formula` objects whose length is equal to the number of regression components in the `mpr` object, e.g., the Weibull model has two components.
`comp`	a numeric vector indicating the regression component(s) to be updated (only needed when `new` is a one-sided `formula`) where "1" $= \lambda$, "2" $= \gamma$ and "3" $= \rho$. For more information on the various components, see `mpr` and `distributions`.
`...`	additional arguments to be passed to the updated `mpr` call.

Details

There are two ways in whcih the `update.mpr` function can be used. The first specificies which component(s) will be updated (via the `comp` argument) along with the update to be applied via `new` which must then be a one-sided `formula`. The second approach specifies both the components in question and the updates to be applied through `new` which is a `list` of one-sided `formula` objects (in this case `comp` is ignored). See "Examples" below.

In the `new` formula (or list of formulae) . means "what is already there".

Value

The fitted, updated `mpr` object.

Author(s)

Kevin Burke.

See Also

`mpr`, `addterm`, `dropterm`, `stepmpr`.

Examples

```
# Veterans' administration lung cancer data
veteran <- survival::veteran
head(veteran)

# Weibull MPR treatment model
mod1 <- mpr(Surv(time, status) ~ list(~ trt, ~ trt), data=veteran,
            family="Weibull")

# remove trt from first component
update(mod1, ~ . - trt, comp=1)
update(mod1, list(~ . - trt, ~ .))

# remove trt from both components
update(mod1, ~ . - trt, comp=1:2)
update(mod1, list(~ . - trt, ~ . - trt))

# add celltype to second component
update(mod1, ~ . + celltype, comp=2)
update(mod1, list(~ . , ~ . + celltype))

# simultaneously remove trt from first component and add celltype to second
# component. This is only possible using the approach where "new" is a list.
update(mod1, list(~ . - trt, ~ . + celltype))

# can also update other things, e.g. "family"
update(mod1, ~ ., family="Gompertz")
update(mod1, ~ . + celltype, family="Loglogistic")

mod2 <- update(mod1, ~ ., family="Burr") # change to Burr model
```

```
mod2
update(mod2, ~ . + celltype, comp=2:3) # add celltype to components 2 and 3
```

Index

addterm, 2, *11, 12*, *15*

Burr (Distributions), 4

Distribution (Distributions), 4
distribution, *8*
distribution (Distributions), 4
Distributions, 4
distributions, *2*, *6–8*, *11*, *14*
distributions (Distributions), 4
dropterm, *11, 12*, *15*
dropterm (addterm), 2

factor, *8*
family (Distributions), 4

Gompertz (Distributions), 4

Loglogistic (Distributions), 4

mpr, *2–5*, 6, *9–15*
mprdists (Distributions), 4

nlm, *6*, *8*

overallpval (summary.mpr), 12

PGW (Distributions), 4
predict.mpr, *7, 8*, 9, *14*
print.mpr (mpr), 6
print.summary.mpr (summary.mpr), 12

stepmpr, *2, 3*, *7, 8*, 10, *15*
summary.mpr, *7–9*, 12

TDL (Distributions), 4

update.mpr, *3*, *12*, 14

Weibull (Distributions), 4
WeibullAFT (Distributions), 4