

Package ‘Formula’

September 23, 2009

Version 0.2-0

Date 2009-09-23

Title Extended Model Formulas

Description Infrastructure for extended formulas with multiple parts on the right-hand side and/or multiple responses on the left-hand side.

Author Yves Croissant, Achim Zeileis

Maintainer Yves Croissant <yves.croissant@let.ish-lyon.cnrs.fr>

Depends R (>= 2.0.0), stats

License GPL-2

Repository CRAN

Date/Publication 2009-09-23 20:48:58

R topics documented:

Formula	1
model.frame.Formula	3

Index	7
--------------	----------

Formula *Extended Formulas: Multiple Responses and Multiple Regressor Parts*

Description

The new class `Formula` extends the base class `formula` by allowing for multiple responses and multiple parts of regressors.

Usage

```
Formula(object)

## S3 method for class 'Formula':
formula(x, lhs = NULL, rhs = NULL,
        collapse = FALSE, update = FALSE, drop = TRUE, ...)

as.Formula(x, ...)
is.Formula(object)
```

Arguments

<code>object, x</code>	an object. For <code>Formula</code> it needs to be a formula object.
<code>lhs, rhs</code>	indexes specifying which elements of the left- and right-hand side, respectively, should be employed. <code>NULL</code> corresponds to all parts, 0 to none.
<code>collapse</code>	logical. Should multiple parts (if any) be collapsed to a single part (essentially by replacing the <code> </code> operator by <code>+</code>)? <code>collapse</code> can be a vector of length 2, corresponding for different handling of left- and right-hand side respectively.
<code>update</code>	logical. Only used if <code>all(collapse)</code> . Should the resulting formula be updated to remove possibly redundant terms occurring in multiple terms?
<code>drop</code>	logical. Should the <code>Formula</code> class be dropped? If <code>TRUE</code> (the default) a formula is returned, if <code>FALSE</code> the corresponding <code>Formula</code> is returned.
<code>...</code>	further arguments.

Details

`Formula` objects extend the basic `formula` objects. These extensions include multi-part formulas such as $y \sim x_1 + x_2 \mid u_1 + u_2 + u_3 \mid v_1 + v_2$, multiple response formulas $y_1 + y_2 \sim x_1 + x_2 + x_3$, multi-part responses such as $y_1 \mid y_2 + y_3 \sim x$, and combinations of these.

The `Formula` creates a `Formula` object from a `formula` which can have the `|` operator on the left- and/or right-hand side (LHS and/or RHS). Essentially, it stores the original `formula` along with attribute lists containing the decomposed parts for the LHS and RHS, respectively.

The main motivation for providing the `Formula` class is to be able to conveniently compute model frames and model matrices or extract selected responses based on an extended formula language. This functionality is provided by methods to the generics `model.frame`, and `model.matrix`. For details and examples, see their manual page: [model.frame.Formula](#).

In addition to these workhorses, a few further methods and functions are provided. By default, the `formula()` method switches back to the original `formula`. Additionally, it allows selection of subsets of the LHS and/or RHS (via `lhs`, and `rhs`) and collapsing multiple parts on the LHS and/or RHS into a single part (via `collapse`).

`is.Formula` checks whether the argument inherits from the `Formula` class.

`as.Formula` is a generic for coercing to `Formula`, the default method first coerces to `formula` and then calls `Formula`.

Methods to further standard generics `print`, `update`, and `length` are provided for `Formula` objects. The latter reports the number of parts on the LHS and RHS, respectively.

Value

Formula returns an object of class Formula which inherits from formula. It is the original formula plus two attributes "lhs" and "rhs" that contain the parts of the decomposed left- and right-hand side, respectively.

See Also

[model.frame.Formula](#)

Examples

```
## create a simple Formula with one response and two regressor parts
f1 <- y ~ x1 + x2 | z1 + z2 + z3
F1 <- Formula(f1)
class(F1)
length(F1)

## switch back to original formula
formula(F1)

## create formula with various transformations
formula(F1, rhs = 1)
formula(F1, collapse = TRUE)
formula(F1, lhs = 0, rhs = 2)

## put it together from its parts
as.Formula(y ~ x1 + x2, ~ z1 + z2 + z3)

## update the formula
update(F1, . ~ . + I(x1^2) | . - z2 - z3)
update(F1, . | y2 + y3 ~ .)

# create a multi-response multi-part formula
f2 <- y1 | y2 + y3 ~ x1 + I(x2^2) | 0 + log(x1) | x3 / x4
F2 <- Formula(f2)
length(F2)

## obtain various subsets using standard indexing
## no lhs, first/second rhs
formula(F2, lhs = 0, rhs = 1:2)
formula(F2, lhs = 0, rhs = -3)
formula(F2, lhs = 0, rhs = c(TRUE, TRUE, FALSE))
## first lhs, third rhs
formula(F2, lhs = c(TRUE, FALSE), rhs = 3)
```

model.frame.Formula

Model Frame/Matrix/Response Construction for Extended Formulas

Description

Computation of model frames, model matrices, and model responses for extended formulas of class Formula.

Usage

```
## S3 method for class 'Formula':
model.frame(formula, data = NULL, ...,
            lhs = NULL, rhs = NULL)
## S3 method for class 'Formula':
model.matrix(object, data = environment(object), ...,
            lhs = NULL, rhs = 1)
## S3 method for class 'Formula':
terms(x, ...,
      lhs = NULL, rhs = NULL)

model.part(object, ...)
## S3 method for class 'Formula':
model.part(object, data, lhs = 0, rhs = 0,
          drop = FALSE, ...)
```

Arguments

<code>formula</code> , <code>object</code> , <code>x</code>	an object of class <code>Formula</code> .
<code>data</code>	a <code>data.frame</code> , list or environment containing the variables in <code>formula</code> . For <code>model.part</code> it needs to be the <code>model.frame</code> .
<code>lhs</code> , <code>rhs</code>	indexes specifying which elements of the left- and right-hand side, respectively, should be employed. <code>NULL</code> corresponds to all parts, <code>0</code> to none. At least one <code>lhs</code> or one <code>rhs</code> has to be specified.
<code>drop</code>	logical. Should the <code>data.frame</code> be dropped for single column data frames?
<code>...</code>	further arguments passed to the respective <code>formula</code> methods.

Details

All three model computations leverage the corresponding standard methods. Additionally, they allow specification of the part(s) of the left- and right-hand side (LHS and RHS) that should be included in the computation.

The idea underlying all three model computations is to extract a suitable `formula` from the more general `Formula` and then calling the standard `model.frame`, `model.matrix`, and `terms` methods.

More specifically, if the `Formula` has multiple parts on the RHS, they are collapsed, essentially replacing `|` by `+`. If there is only a single response on the LHS, then it is kept on the LHS. Otherwise all parts of the formula are collapsed on the RHS (because `formula` objects can not have multiple responses). Hence, for multi-response `Formula` objects, the (non-generic) `model.response` does not give the correct results. To avoid confusion a new generic `model.part` with suitable `formula` method is provided which can always be used instead of `model.response`. Note,

however, that it has a different syntax: It requires the `Formula` object in addition to the readily processed `model.frame` supplied in `data` (and optionally the `lhs`). Also, it returns either a `data.frame` with multiple columns or a single column (dropping the `data.frame` property) depending on whether multiple responses are employed or not.

See Also

[Formula](#), [model.frame](#), [model.matrix](#), [terms](#), [model.response](#)

Examples

```
## artificial example data
set.seed(1090)
dat <- as.data.frame(matrix(round(runif(21), digits = 2), ncol = 7))
colnames(dat) <- c("y1", "y2", "y3", "x1", "x2", "x3", "x4")
for(i in c(2, 6:7)) dat[[i]] <- factor(dat[[i]] > 0.5, labels = c("a", "b"))
dat$y2[1] <- NA
dat

#####
## single response and two-part RHS ##
#####

## single response with two-part RHS
F1 <- Formula(log(y1) ~ x1 + x2 | I(x1^2))
length(F1)

## set up model frame
mf1 <- model.frame(F1, data = dat)
mf1

## extract single response
model.part(F1, data = mf1, lhs = 1, drop = TRUE)
model.response(mf1)
## model.response() works as usual

## extract model matrices
model.matrix(F1, data = mf1, rhs = 1)
model.matrix(F1, data = mf1, rhs = 2)

#####
## multiple responses and multiple RHS ##
#####

## set up Formula
F2 <- Formula(y1 + y2 | log(y3) ~ x1 + I(x2^2) | 0 + log(x1) | x3 / x4)
length(F2)

## set up full model frame
mf2 <- model.frame(F2, data = dat)
mf2
```

```
## extract responses
model.part(F2, data = mf2, lhs = 1)
model.part(F2, data = mf2, lhs = 2)
## model.response(mf2) does not give correct results!

## extract model matrices
model.matrix(F2, data = mf2, rhs = 1)
model.matrix(F2, data = mf2, rhs = 2)
model.matrix(F2, data = mf2, rhs = 3)

#####
## misc ##
#####

## Formulas with '.'
F3 <- Formula(y1 | y2 ~ .)
mf3 <- model.frame(F3, data = dat)
## without y1 or y2
model.matrix(F3, data = mf3)
## without y1 but with y2
model.matrix(F3, data = mf3, lhs = 1)
## without y2 but with y1
model.matrix(F3, data = mf3, lhs = 2)
```

Index

*Topic **classes**

Formula, 1

*Topic **models**

model.frame.Formula, 3

as.Formula (*Formula*), 1

Formula, 1, 5

formula, 1

formula.Formula (*Formula*), 1

is.Formula (*Formula*), 1

length, 2

length.Formula (*Formula*), 1

model.frame, 2, 4, 5

model.frame.Formula, 2, 3, 3

model.matrix, 2, 4, 5

model.matrix.Formula

(*model.frame.Formula*), 3

model.part (*model.frame.Formula*),
3

model.response, 4, 5

print, 2

print.Formula (*Formula*), 1

terms, 4, 5

terms.Formula

(*model.frame.Formula*), 3

update, 2

update.Formula (*Formula*), 1