Package ‘Formula’

October 12, 2022

Version 1.2-4

Date 2020-10-16

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 (see <doi:10.18637/jss.v034.i01>).

Depends R (>= 2.0.0), stats

License GPL-2 | GPL-3

NeedsCompilation no

Author Achim Zeileis [aut, cre] (<https://orcid.org/0000-0003-0918-3766>), Yves Croissant [aut]

Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>

Repository CRAN

Date/Publication 2020-10-16 13:50:06 UTC

R topics documented:

`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

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

Details

`Formula` objects extend the basic `formula` objects. These extensions include multi-part formulas such as `y ~ x1 + x2 | u1 + u2 + u3 | v1 + v2`, multiple response formulas `y1 + y2 ~ x1 + x2 + x3`, multi-part responses such as `y1 | y2 + y3 ~ 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`. The default and `formula` method also take an optional `env` argument, specifying the environment of the resulting `Formula`. In the latter case, this defaults to the environment of the `formula` supplied.

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.

References

Zeileis A, Croissant Y (2010). Extended Model Formulas in R: Multiple Parts and Multiple Responses. *Journal of Statistical Software*, **34**(1), 1–13. doi: 10.18637/jss.v034.i01

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/seconde 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, dot = "separate")
## S3 method for class 'Formula'
model.matrix(object, data = environment(object), ...,
  lhs = NULL, rhs = 1, dot = "separate")
## S3 method for class 'Formula'
terms(x, ...,
  lhs = NULL, rhs = NULL, dot = "separate")

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

Arguments

`formula, object, x`	an object of class `Formula`.
`data`	a data.frame, list or environment containing the variables in `formula`. For `model.part` it needs to be the `model.frame`.
`lhs, rhs`	indexes specifying which elements of the left- and right-hand side, respectively, should be employed. `NULL` corresponds to all parts, `0` to none. At least one `lhs` or one `rhs` has to be specified.
`dot`	character specifying how to process formula parts with a dot (`.`) on the right-hand side. This can be: `"separate"` so that each formula part is expanded separately. `"sequential"` so that the parts are expanded sequentially conditional on all prior parts. `"previous"` so the part is expanded to the previous part.
`drop`	logical. Should the `data.frame` be dropped for single column data frames?
`terms`	logical. Should the `"terms"` attribute (corresponding to the `model.part` extracted) be added?
`...`	further arguments passed to the respective `formula` 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.

If the the formula contains one or more dots (`.`), some care has to be taken to process these correctly, especially if the LHS contains transformartions (such as `log`, `sqrt`, `cbind`, `Surv`, etc.). Calling the `terms` method with the original data (untransformed, if any) resolves all dots (by default separately for each part, otherwise sequentially) and also includes the original and updated formula as part of the terms. When calling `model.part` either the original untransformed data should be provided along with a `dot` specification or the transformed `model.frame` from the same formula without another `dot` specification (in which case the `dot` is inferred from the `terms` of the `model.frame`).

References

Zeileis A, Croissant Y (2010). Extended Model Formulas in R: Multiple Parts and Multiple Responses. *Journal of Statistical Software*, **34**(1), 1–13. doi: 10.18637/jss.v034.i01

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)

#######################
## Formulas with '.' ##
#######################

## set up Formula with a single '.'
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)

## set up Formula with multiple '.'
F3 <- Formula(y1 | y2 | log(y3) ~ . - x3 - x4 | .)
```

```
## process both '.' separately (default)
mf3 <- model.frame(F3, data = dat, dot = "separate")
## only x1-x2
model.part(F3, data = mf3, rhs = 1)
## all x1-x4
model.part(F3, data = mf3, rhs = 2)

## process the '.' sequentially, i.e., the second RHS conditional on the first
mf3 <- model.frame(F3, data = dat, dot = "sequential")
## only x1-x2
model.part(F3, data = mf3, rhs = 1)
## only x3-x4
model.part(F3, data = mf3, rhs = 2)

## process the second '.' using the previous RHS element
mf3 <- model.frame(F3, data = dat, dot = "previous")
## only x1-x2
model.part(F3, data = mf3, rhs = 1)
## x1-x2 again
model.part(F3, data = mf3, rhs = 2)

##############################
## Process multiple offsets ##
##############################

## set up Formula
F4 <- Formula(y1 ~ x3 + offset(x1) | x4 + offset(log(x2)))
mf4 <- model.frame(F4, data = dat)
## model.part can be applied as above and includes offset!
model.part(F4, data = mf4, rhs = 1)
## additionally, the corresponding corresponding terms can be included
model.part(F4, data = mf4, rhs = 1, terms = TRUE)
## hence model.offset() can be applied to extract offsets
model.offset(model.part(F4, data = mf4, rhs = 1, terms = TRUE))
model.offset(model.part(F4, data = mf4, rhs = 2, terms = TRUE))
```

Index