

0.1 `parse.formula`: Parsing user-input formulas into multiple syntax

Description

Parse the input formula (or list of formulas) into the standard format described below. Since labels for this format will vary by model, `parse.formula` will evaluate a function `describe.model`, where `model` is given as an input to `parse.formula`.

If the `describe.model` function has more than one parameter for which `ExpVar = TRUE` and `DepVar = TRUE`, then the user-specified equations must have labels to match those parameters, else `parse.formula` should return an error. In addition, if the formula entries are not unambiguous, then `parse.formula` returns an error.

Usage

```
parse.formula(formula, model, data = NULL)
```

Arguments

<code>formula</code>	either a single formula or a list of <code>formula</code> objects
<code>model</code>	a character string specifying the name of the model
<code>data</code>	an optional data frame for models that require a factor response variable

Details

Acceptable user inputs are as follows:

	User Input	Output from <code>parse.formula</code>
Same covariates, separate effects	<code>cbind(y1, y2) x1 + x2 * x3</code>	<code>list(mu1 = y1 x1 + x2 * x3, mu2 = y2 x1 + x2 * x3, rho = 1)</code>
With <code>rho</code> as a systematic equation	<code>list(cbind(y1, y2) x1 + x2, rho = x4 + x5)</code>	<code>list(mu1 = y1 x1 + x2, mu2 = y2 x1 + x2, rho = x4 + x5)</code>
With constraints (same variable)	<code>list(mu1 = y1 x1 + tag(x2, "x2"), mu2 = y2 x3 + tag(x2, "x2"))</code>	<code>list(mu1 = y1 x1 + tag(x2, "x2"), mu2 = y2 x3 + tag(x2, "x2"), rho = 1)</code>
With constraints (different variables)	<code>list(mu1 = y1 x1 + tag(x2, "z1"), mu2 = y2 x3 + tag(x4, "z1"))</code>	<code>list(mu1 = y1 x1 + tag(x2, "z1"), mu2 = y2 x3 + tag(x4, "z1"), rho = 1)</code>

Value

The output is a list of formula objects with class `c("multiple", "list")`. Let's say that the name of the model is `"bivariate.probit"`, and the corresponding describe function is `describe.bivariate.probit`, which identifies `mu1` and `mu2` as systematic components, and an ancillary parameter `rho`, which may be parameterized, but is estimated as a scalar by default.

Author(s)

Kosuke Imai [j\(kimai@princeton.edu\)¿](mailto:kimai@princeton.edu); Gary King [j\(king@harvard.edu\)¿](mailto:king@harvard.edu); Olivia Lau [j\(olau@fas.harvard.edu\)¿](mailto:olau@fas.harvard.edu); Ferdinand Alimadhi [j\(falimadhi@iq.harvard.edu\)¿](mailto:falimadhi@iq.harvard.edu)

See Also

`parse.par`, `model.frame.multiple`, `model.matrix.multiple`, and the full Zelig manual at <http://gking.harvard.edu/zelig>.

Examples

```
## Not run:
data(sanction)
formulae <- list(cbind(import, export) ~ coop + cost + target)
fml <- parse.formula(formulae, model = "bivariate.probit")
D <- model.frame(fml, data = sanction)
## End(Not run)
```