0.1 `describe`: Describe a model's systematic and stochastic parameters

Description

In order to use `parse.formula()`, `parse.par()`, and the `model.*.multiple()` commands, you must write a `describe.mymodel()` function where `mymodel` is the name of your modeling function. (Hence, if your function is called `normal.regression()`, you need to write a `describe.normal.regression()` function.) Note that `describe()` is *not* a generic function, but is called by `parse.formula(..., model = "mymodel")` using a combination of `paste()` and `exists()`. You will never need to call `describe.mymodel()` directly, since it will be called from `parse.formula()` as that function checks the user-input formula or list of formulas.

Syntax

```
describe.mymodel()
```

Arguments

The `describe.mymodel()` function takes no arguments.

Output Values

The `describe.mymodel()` function returns a list with the following information:

- `category`: a character string, consisting of one of the following:
  - `"continuous"`: the dependent variable is continuous, numeric, and unbounded (e.g., normal regression), but may be censored with an associated censoring indicator (e.g., tobit regression).
  - `"dichotomous"`: the dependent variable takes two discrete integer values, usually 0 and 1 (e.g., logistic regression).
  - `"ordinal"`: the dependent variable is an ordered factor response, taking 3 or more discrete values which are arranged in an ascending or descending manner (e.g., ordered logistic regression).
  - `"multinomial"`: the dependent variable is an unordered factor response, taking 3 or more discrete values which are arranged in no particular order (e.g., multinomial logistic regression).
  - `"count"`: the dependent variable takes integer values greater than or equal to 0 (e.g., Poisson regression).

  - `"bounded"`: the dependent variable is a continuous numeric variable, that is restricted (bounded within) some range (e.g., $(0, \infty)$). The variable may also be censored either on the left and/or right side, with an associated censoring indicator (e.g., Weibull regression).
  - `"mixed"`: the dependent variables are a mix of the above categories (usually applies to multiple equation models).

  Selecting the category is particularly important since it sets certain interface parameters for the entire GUI.

- `package`: (optional) a list with the following elements

  - `name`: a characters string with the name of the package containing the `mymodel()` function.
  - `version`: the minimum version number that works with Zelig.
  - `CRAN`: if the package is not hosted on CRAN mirrors, provide the URL here as a character string. You should be able to install your package from this URL using `name`, `version`, and `CRAN`:

    ```
    install.packages(name, repos = CRAN, installWithVers = TRUE)
    ```

    By default, `CRAN = "http://cran.us.r-project.org/"`.

- `parameters`: For each systematic and stochastic parameter (or set of parameters) in your model, you should create a list (named after the parameters as given in your model's notation, e.g., `mu`, `sigma`, `theta`, etc.; not literally `myparameter`) with the following information:

  - `equations`: an integer number of equations for the parameter. For parameters that can take an undefined number of equations (for example in seemingly unrelated regression), use `c(2, Inf)` or `c(2, 999)` to indicate that the parameter can take a minimum of two equations up to a theoretically infinite number of equations.
  - `tagsAllowed`: a logical value (`TRUE`/`FALSE`) specifying whether a given parameter allows constraints. If there is only one equation for a parameter (for example, `mu` for the normal regression model has `equations = 1`), then `tagsAllowed = FALSE` by default. If there are two or more equations for the parameter (for example, `mu` for the bivariate probit model has `equations = 2`), then `tagsAllowed = TRUE` by default.
  - `depVar`: a logical value (`TRUE`/`FALSE`) specifying whether a parameter requires a corresponding dependent variable.

- `expVar`: a logical value (`TRUE`/`FALSE`) specifying whether a parameter allows explanatory variables. If `depVar = TRUE` and `expVar = TRUE`, we call the parameter a "systematic component" and `parse.formula()` will fail if formula(s) are not specified for this parameter. If `depVar = FALSE` and `expVar = TRUE`, the parameter is estimated as a scalar ancillary parameter, with default formula `~ 1`, if the user does not specify a formula explicitly. If `depVar = FALSE` and `expVar = FALSE`, the parameter can only be estimated as a scalar ancillary parameter.
- `specialFunction`: (optional) a character string giving the name of a function that appears on the left-hand side of the formula. Options include `"Surv"`, `"cbind"`, and `"as.factor"`.
- `varInSpecial`: (optional) a scalar or vector giving the number of variables taken by the `specialFunction`. For example, `Surv()` takes a minimum of 2 arguments, and a maximum of 4 arguments, which is represented as `c(2, 4)`.

If you have more than one parameter (or set of parameters) in your model, you will need to produce a `myparameter` list for each one. See examples below for details.

Examples

For a Normal regression model with mean `mu` and scalar variance parameter `sigma2`, the minimal `describe.*()` function is as follows:

```
describe.normal.regression <- function() {
  category <- "continuous"
  mu <- list(equations = 1,              # Systematic component
             tagsAllowed = FALSE,
             depVar = TRUE,
             expVar = TRUE)
  sigma2 <- list(equations = 1,          # Scalar ancillary parameter
                 tagsAllowed = FALSE,
                 depVar = FALSE,
                 expVar = FALSE)
  pars <- list(mu = mu, sigma2 = sigma2)
  model <- list(category = category, parameters = pars)
}
```

See Section **??** for full code to execute this model from scratch in R with Zelig.

Now consider a bivariate probit model with parameter vector `mu` and correlation parameter `rho` (which may or may not take explanatory variables). Since the bivariate probit function uses the `pmvnorm()` function from the mvtnorm library, we list this under `package`.

```
describe.bivariate.probit <- function() {
  category <- "dichotomous"
  package <- list(name = "mvtnorm",
```

```
                  version = "0.7")
  mu <- list(equations = 2,               # Systematic component
             tagsAllowed = TRUE,
             depVar = TRUE,
             expVar = TRUE)
  rho <- list(equations = 1,              # Optional systematic component
             tagsAllowed = FALSE,         #   Estimated as an ancillary
             depVar = FALSE,              #   parameter by default
             expVar = TRUE)
  pars <- list(mu = mu, rho = rho)
  list(category = category, package = package, parameters = pars)
}
```

See Section **??** for the full code to write this model from scratch in R with Zelig.

For a multinomial logit model, which takes an undefined number of equations (corresponding to each level in the response variable):

```
describe.multinomial.logit <- function() {
  category <- "multinomial"
  mu <- list(equations = c(1, Inf),
             tagsAllowed = TRUE,
             depVAR = TRUE,
             expVar = TRUE,
             specialFunction <- "as.factor",
             varInSpecial <- c(1, 1))
  list(category = category, parameters = list(mu = mu))
}
```

(This example does not have corresponding executable sample code.)

See Also

- Section **??** for an overview of how the `describe.*()` function works with `parse.formula()`.
- Section **??** for information on `parse.formula()`.

Contributors

Kosuke Imai, Gary King, Olivia Lau, and Ferdinand Alimadhi.