Package: enrichwith (via r-universe)

August 30, 2024

Title Methods to Enrich R Objects with Extra Components

Version 0.3

Description Provides the ``enrich'' method to enrich list-like R objects with new, relevant components. The current version has methods for enriching objects of class 'family', 'link-glm', 'lm', 'glm' and 'betareg'. The resulting objects preserve their class, so all methods associated with them still apply. The package also provides the 'enriched_glm' function that has the same interface as 'glm' but results in objects of class 'enriched_glm'. In addition to the usual components in a `glm` object, 'enriched_glm' objects carry an object-specific simulate method and functions to compute the scores, the observed and expected information matrix, the first-order bias, as well as model densities, probabilities, and quantiles at arbitrary parameter values. The package can also be used to produce customizable source code templates for the structured implementation of methods to compute new components and enrich arbitrary objects.

Depends R (>= 3.0.0)

URL https://github.com/ikosmidis/enrichwith

BugReports https://github.com/ikosmidis/enrichwith/issues

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests whisker, SuppDists, brglm, ggplot2, knitr, rmarkdown

Enhances betareg, gnm, stats

VignetteBuilder knitr

Repository https://ikosmidis.r-universe.dev

RemoteUrl https://github.com/ikosmidis/enrichwith

RemoteRef HEAD

RemoteSha b78fbdcbd6a843479ff1596b6a56c78205f8b6a2

Contents

coef.enriched_glm *Function to extract model coefficients from objects of class* enriched_glm

Description

Function to extract model coefficients from objects of class enriched_glm

Usage

```
## S3 method for class 'enriched_glm'
coef(object, model = c("mean", "full",
  "dispersion"), ...)
```

Arguments

object	an object of class enriched_glm
model	either "mean" for the estimates of the parameters in the linear predictor, or "dispersion" for the estimate of the dispersion, or "full" for all estimates
...	currently unused

coef.enriched_lm *Function to extract model coefficients from objects of class* enriched_lm

Description

Function to extract model coefficients from objects of class enriched_lm

Usage

```
## S3 method for class 'enriched_lm'
coef(object, model = c("mean", "full",
  "dispersion"), ...)
```

Arguments

object	an object of class enriched_lm
model	either "mean" for the estimates of the parameters in the linear predictor, or "dispersion" for the estimate of the dispersion, or "full" for all estimates
...	currently unused

`create_enrichwith_skeleton` *Create a enrichwith skeleton*

Description

Create an enrichwith skeleton file for the structured implementation of methods to compute new components for objects of a specific class

Usage

```
create_enrichwith_skeleton(class, option, description, component, path,
  filename = paste0(class, "_options.R"), attempt_rename = TRUE)
```

Arguments

`class`	the class of the objects to be enriched
`option`	a character vector with components the enrichment options
`description`	a character vector of length `length(options)` with components the description of the enrichment options
`component`	a list of as many character vectors as `length(option)`, specifying the names of the components that each option will add to the object after enrichment
`path`	the path where the skeleton file will be created
`filename`	the name of the skeleton file
`attempt_rename`	attempt to rename syntactically incorrect component names? Default is `TRUE`

Value

A file with the necessary functions to use `enrichwith` infrastructure. The skeleton consists of the following functions

- One `compute_component.class` function per component name from `unique(unlist(component))`. The function takes as input the object to be enriched and returns as output the component to be added to the object.
- The `get_enrichment_options.class` function, that takes as input the object to be enriched and an enrichment option, and returns the names of the components that will be appended to the object for this option. This function can also be used to list the available options and print their description.
- The `enrich.class` function

Examples

```
## Not run:
# Set the directory where the skeleton is placed
my_path <- "~/Downloads"
# This is the call that created the enrichment skeleton for glms
# that ships with the package
create_enrichwith_skeleton(class = "glm",
      option = c("auxiliary functions", "score vector",
       "mle of dispersion", "expected information",
       "observed information", "first-order bias"),
      description = c("various likelihood-based quantities
        (gradient of the log-likelihood, expected and observed
        information matrix and first term in the expansion of
        the bias of the mle) and a simulate method as functions
        of the model parameters",
       "gradient of the log-likelihood at the mle",
       "mle of the dispersion parameter",
       "expected information matrix evaluated at the mle",
       "observed information matrix evaluated at the mle",
       "first term in the expansion of the bias of the mle
        at the mle"),
      component = list("auxiliary_functions", "score_mle",
                       "dispersion_mle",
                       "expected_information_mle",
                       "observed_information_mle",
                       "bias_mle"),
      path = my_path,
      attempt_rename = FALSE)


## End(Not run)
```

`endometrial` *Histology grade and risk factors for 79 cases of endometrial cancer*

Description

Histology grade and risk factors for 79 cases of endometrial cancer

Usage

```
endometrial
```

Format

A data frame with 79 rows and 4 variables:

NV neovasculization with coding 0 for absent and 1 for present

PI pulsality index of arteria uterina

EH endometrium heigh

HG histology grade with coding 0 for low grade and 1 for high grade

Source

The packaged data set was downloaded in `.dat` format from http://www.stat.ufl.edu/~aa/glm/data. The latter link provides the data sets used in Agresti (2015).

The endometrial data set was first analysed in Heinze and Schemper (2002), and was originally provided by Dr E. Asseryanis from the Medical University of Vienna.

Agresti, A. 2015. *Foundations of Linear and Generalized Linear Models*. Wiley Series in Probability and Statistics. Wiley.

Heinze, G., and M. Schemper. 2002. "A Solution to the Problem of Separation in Logistic Regression." *Statistics in Medicine* 21:2409–19.

`enrich` *Generic method for enriching objects*

Description

Generic method for enriching objects

Usage

```
enrich(object, with, ...)
```

Arguments

`object`	the object to be enriched
`with`	a character vector with enrichment options for `object`
`...`	Arguments to be passed to other methods

See Also

`enrich.glm`, `enriched_glm`, `enrich.family`, `enrich.link-glm`, `enrich.betareg`

enrich.betareg *Enrich objects of class betareg*

Description

Enrich objects of class betareg with any or all of a set of auxiliary functions, the expected information at the maximum likelihood estimator, and the first term in the expansion of the bias of the maximum likelihood estimator.

Usage

```
## S3 method for class 'betareg'
enrich(object, with = "all", ...)
```

Arguments

object	an object of class betareg
with	a character vector of options for the enrichment of object
...	extra arguments to be passed to the compute_* functions

Details

The auxiliary_functions component consists of any or all of the following functions:

- score: the log-likelihood derivatives as a function of the model parameters; see get_score_function.betareg
- information: the expected information as a function of the model parameters; see get_information_function.beta
- bias: the first-order term in the expansion of the bias of the maximum likelihood estimator as a function of the model parameters; see get_bias_function.betareg
- simulate: a simulate function for betareg objects that can simulate variates from the model at user-supplied parameter values for the regression parameters (default is the maximum likelihood estimates); see get_simulate_function.betareg

Value

The object object of class betareg with extra components. get_enrichment_options.betareg() returns the components and their descriptions.

Examples

```
## Not run:
if (require("betareg")) {

   data("GasolineYield", package = "betareg")
   gy <- betareg(yield ~ batch + temp, data = GasolineYield)

  # Get a function that evaluates the expected information for gy at supplied parameter values
   gy_info <- get_information_function(gy)
```

```
.   # compare standard errors with what `summary` returns
   all.equal(sqrt(diag(solve(gy_info()))[1:11],
             coef(summary(gy))$mean[, 2], check.attributes = FALSE)
.   # evaluating at different parameter values
   gy_info(rep(1, length = 12))

   # Get a function that evaluates the first-order bias of gy at supplied parameter values
   gy_bias <- get_bias_function(gy)
   # compare with internal betareg implementation of bias correction
   gy_bc <- update(gy, type = "BC")
   all.equal(gy_bias(coef(gy)), gy_bc$bias, check.attributes = FALSE)

 }

## End(Not run)
```

`enrich.family` *Enrich objects of class* `family`

Description

Enrich objects of class `family` with family-specific mathematical functions

Usage

```
## S3 method for class 'family'
enrich(object, with = "all", ...)
```

Arguments

`object`	an object of class `family`
`with`	a character vector with enrichment options for `object`
`...`	extra arguments to be passed to the `compute_*` functions

Details

`family` objects specify characteristics of the models used by functions such as `glm`. The families implemented in the `stats` package include `binomial`, `gaussian`, `Gamma`, `inverse.gaussian`, and `poisson`, which are all special cases of the exponential family of distributions that have probability mass or density function of the form

$$f(y;\theta,\phi) = \exp\left\{\frac{y\theta - b(\theta) - c_1(y)}{\phi/m} - \frac{1}{2}a\left(-\frac{m}{\phi}\right) + c_2(y)\right\} \quad y \in Y \subset \Re\,, \theta \in \Theta \subset \Re\,, \phi > 0$$

where $m > 0$ is an observation weight, and $a(.)$, $b(.)$, $c_1(.)$ and $c_2(.)$ are sufficiently smooth, real-valued functions.

The current implementation of `family` objects includes the variance function (`variance`), the deviance residuals (`dev.resids`), and the Akaike information criterion (`aic`). See, also `family`.

The enrich method can further enrich exponential family distributions with θ in terms of μ (theta), the functions $b(\theta)$ (bfun), $c_1(y)$ (c1fun), $c_2(y)$ (c2fun), $a(\zeta)$ (fun), the first two derivatives of $V(\mu)$ (d1variance and d2variance, respectively), and the first four derivatives of $a(\zeta)$ (d1afun, d2afun, d3afun, d4afun, respectively).

Corresponding enrichment options are also avaialble for quasibinomial, quasipoisson and wedderburn families.

The quasi families are enriched with d1variance and d2variance.

See enrich.link-glm for the enrichment of link-glm objects.

Value

The object object of class family with extra components. get_enrichment_options.family() returns the components and their descriptions.

See Also

enrich.link-glm, make.link

Examples

```
## An example from ?glm to illustrate that things still work with
## enriched families
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family = enrich(poisson()))
anova(glm.D93)
summary(glm.D93)
```

---

enrich.glm *Enrich objects of class* glm

Description

Enrich objects of class glm with any or all of a set of auxiliary functions, the maximum likelihood estimate of the dispersion parameter, the expected or observed information at the maximum likelihood estimator, and the first term in the expansion of the bias of the maximum likelihood estimator.

Usage

```
## S3 method for class 'glm'
enrich(object, with = "all", ...)
```

Arguments

object	an object of class glm
with	a character vector of options for the enrichment of object
...	extra arguments to be passed to the compute_* functions

Details

The auxiliary_functions component consists of any or all of the following functions:

- score: the log-likelihood derivatives as a function of the model parameters; see get_score_function.glm
- information: the expected or observed information as a function of the model parameters; see get_information_function.glm
- bias: the first-order term in the expansion of the bias of the maximum likelihood estimator as a function of the model parameters; see get_bias_function.glm
- simulate: a simulate function for glm objects that can simulate variates from the model at user-supplied parameter values for the regression parameters and the dispersion (default is the maximum likelihood estimates); see get_simulate_function.glm
- dmodel: computes densities or probability mass functions under the model at user-supplied data.frames and at user-supplied values for the regression parameters and the dispersion, if any (default is at the maximum likelihood estimates); see get_dmodel_function.glm
- pmodel: computes distribution functions under the model at user-supplied data.frames and at user-supplied values for the regression parameters and the dispersion, if any (default is at the maximum likelihood estimates); see get_pmodel_function.glm
- qmodel: computes quantile functions under the model at user-supplied data.frames and at user-supplied values for the regression parameters and the dispersion, if any (default is at the maximum likelihood estimates); see get_qmodel_function.glm

Value

The object object of class glm with extra components. See get_enrichment_options.glm() for the components and their descriptions.

Examples

```
## Not run:
# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
clotting <- data.frame(
   u = c(5,10,15,20,30,40,60,80,100, 5,10,15,20,30,40,60,80,100),
   time = c(118,58,42,35,27,25,21,19,18,69,35,26,21,18,16,13,12,12),
   lot = factor(c(rep(1, 9), rep(2, 9))))
cML <- glm(time ~ lot*log(u), data = clotting, family = Gamma)

# The simulate method for the above fit would simulate at coef(cML)
# for the regression parameters and MASS::gamma.dispersion(cML) for
# the dispersion. It is not possible to simulate at different
# parameter values than those, at least not, without "hacking" the
# cML object.

# A general simulator for cML results via its enrichment with
# auxiliary functions:
cML_functions <- get_auxiliary_functions(cML)
# which is a shorthand for
# enriched_cML <- enrich(cML, with = "auxiliary functions")
# cML_functions <- enriched_cML$auxiliary_functions
```

```
# To simulate 2 samples at the maximum likelihood estimator do
dispersion_mle <- MASS::gamma.dispersion(cML)
cML_functions$simulate(coef = coef(cML),
                       dispersion = dispersion_mle,
                       nsim = 2, seed = 123)
# To simulate 5 samples at c(0.1, 0.1, 0, 0) and dispersion 0.2 do
cML_functions$simulate(coef = c(0.1, 0.1, 0, 0),
                       dispersion = 0.2,
                       nsim = 5, seed = 123)


## End(Not run)

## Not run:

## Reproduce left plot in Figure 4.1 in Kosimdis (2007)
## (see http://www.ucl.ac.uk/~ucakiko/files/ikosmidis_thesis.pdf)
mod <- glm(1 ~ 1, weights = 10, family = binomial())
enriched_mod <- enrich(mod, with = "auxiliary functions")
biasfun <- enriched_mod$auxiliary_functions$bias
probabilities <- seq(1e-02, 1 - 1e-02, length = 100)
biases <- Vectorize(biasfun)(qlogis(probabilities))
plot(probabilities, biases, type = "l", ylim = c(-0.5, 0.5),
     xlab = expression(pi), ylab = "first-order bias")
abline(h = 0, lty = 2); abline(v = 0.5, lty = 2)
title("First-order bias of the MLE of the log-odds", sub = "m = 10")

## End(Not run)
```

`enrich.link-glm` *Enrich objects of class* `link-glm`

Description

Enrich objects of class `link-glm` with further derivatives of `linkinv` with respect to `eta`.

Usage

```
## S3 method for class '`link-glm`'
enrich(object, with = "all", ...)
```

Arguments

`object`	an object of class `link-glm`
`with`	a character vector with enrichment options for `object`
`...`	extra arguments to be passed to the `compute_*` functions

Details

The enrich.link-glm method supports logit, probit, cauchit, cloglog, identity, log, sqrt, 1/mu^2, inverse, as well as the power family of links.

Value

The object object of class link-glm with extra components. get_enrichment_options.link-glm() returns the components and their descriptions.

Examples

```
# Example
elogit <- enrich(make.link("logit"), with = "inverse link derivatives")
str(elogit)
elogit$d2mu.deta
elogit$d3mu.deta
```

---

enrich.lm *Enrich objects of class lm*

Description

Enrich objects of class lm with any or all of a set auxiliary functions, the maximum likelihood estimate of the dispersion parameter, the expected or observed information at the maximum likelihood estimator, and the first term in the expansion of the bias of the maximum likelihood estimator.

Usage

```
## S3 method for class 'lm'
enrich(object, with = "all", ...)
```

Arguments

object	an object of class lm
with	a character vector of options for the enrichment of object
...	extra arguments to be passed to the compute_* functions

Details

The auxiliary functions consist of the score functions, the expected or observed information, the first-order bias of the maximum likelihood estimator as functions of the model parameters, and a simulate function that takes as input the model parameters (including the dispersion if any). The result from the simulate auxiliary function has the same structure to that of the simulate method for lm objects.

Value

The object object of class lm with extra components. get_enrichment_options.lm() returns the components and their descriptions.

enriched_glm *Fitting generalized linear models enriched with useful components*

Description

enriched_glm fits generalized linear models using glm and then enriches the resulting object with all enrichment options.

Usage

```
enriched_glm(formula, family = gaussian, ...)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
...	other arguments passed to glm

Details

enriched_glm has the same interface as glm

Value

An object of class enriched_glm that contains all the components of a glm object, along with a set of auxiliary functions (score function, information matrix, a simulate method, first term in the expansion of the bias of the maximum likelihood estimator, and dmodel, pmodel, qmodel), the maximum likelihood estimate of the dispersion parameter, the expected or observed information at the maximum likelihood estimator, and the first term in the expansion of the bias of the maximum likelihood estimator.

See enrich.glm for more details and links for the auxiliary functions.

Examples

```
## Not run:
# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
clotting <- data.frame(
   u = c(5,10,15,20,30,40,60,80,100, 5,10,15,20,30,40,60,80,100),
   time = c(118,58,42,35,27,25,21,19,18,69,35,26,21,18,16,13,12,12),
   lot = factor(c(rep(1, 9), rep(2, 9))))

# Fit a generalized linear model
```

```
cML <- enriched_glm(time ~ lot*log(u), data = clotting, family = Gamma("log"))

# Evaluate the densities at the data points in clotting at the
# maximum likelihood estimates
cML_dmodel <- get_dmodel_function(cML) # same as cML$auxiliary_functions$dmodel
cML_dmodel()

# Evaluate the densities at supplied data points
new_data <- data.frame(u = c(15:17, 15:17),
                       time = c(30:32, 15:17),
                       lot = factor(c(1, 1, 1, 2, 2, 2)))
cML_dmodel(data = new_data)

# Get pmodel and qmodel function
cML_pmodel <- get_pmodel_function(cML) # same as cML$auxiliary_functions$pmodel
cML_qmodel <- get_qmodel_function(cML) # same as cML$auxiliary_functions$qmodel

# The following should return c(30:32, 15:17)
probs <- cML_pmodel(data = new_data)
cML_qmodel(probs, data = new_data)

# Evaluate the observed information matrix at the MLE
cML_info <- get_information_function(cML)
cML_info(type = "observed")

# Wald tests based on the observed information at the
# moment based esimator of the dispersion
dispersion <- summary(cML)$dispersion
cML_vcov_observed <- solve(cML_info(dispersion = dispersion, type = "observed"))
lmtest::coeftest(cML, vcov = cML_vcov_observed)

# Wald tests based on the expected information at the
# moment based esimator of the dispersion
cML_vcov_expected <- solve(cML_info(dispersion = dispersion, type = "expected"))
lmtest::coeftest(cML, vcov = cML_vcov_expected)
# Same statistics as coef(summary(cML))[, "t value"]


## End(Not run)
```

---

enrichwith *Methods to enrich list-like R objects with extra components*

Description

The enrichwith package provides the enrich method to enrich list-like R objects with new, relevant components. The resulting objects preserve their class, so all methods associated with them still apply. The package can also be used to produce customisable source code templates for the structured implementation of methods to compute new components

Details

Depending on the object, enriching it can be a tedious task. The **enrichwith** package aims to streamline the task into 3 simple steps:

1. Use `create_enrichwith_skeleton` to produce a customisable enrichwith template.
2. Edit the `compute_*` functions by adding the specific code that calculates the components.
3. Finalise the documentation and/or include more examples.

The first step results in a template that includes all necessary functions to carry out the enrichment. The second step is where the user edits the template and implements the calculation of the components that the object will be enriched with. Specifically, each `compute_*` function takes as input the object to be enriched and returns the corresponding new component to be added to the object.

Everything else (for example, mapping between the enrichment options and the components that the enriched object will have, checks that an enrichment option exists, listing enrichment options, enriching the object, and so on) is taken care of by the methods in **enrichwith**.

Developers can either put their enrichwith templates in their packages or are welcome to contribute their template to enrichwith, particularly if that extends core R objects.

See Also

`enrich.glm` and `enriched_glm`, `enrich.family`, `enrich.link-glm`, `enrich.betareg`

---

get_auxiliary_functions

Generic method for extracting or computing auxiliary functions for objects

---

Description

Generic method for extracting or computing auxiliary functions for objects

Usage

```
get_auxiliary_functions(object, ...)
```

Arguments

`object`	the object to be enriched or the enriched object
`...`	currently not used

`get_auxiliary_functions.betareg` *Function to compute/extract auxiliary functions from objects of class* `betreg/enriched_betareg`

Description

Function to compute/extract auxiliary functions from objects of class `betreg/enriched_betareg`

Usage

```
## S3 method for class 'betareg'
get_auxiliary_functions(object, ...)
```

Arguments

`object`	an object of class `betareg` or`enriched_betareg`
`...`	currently not used

Details

See `enrich.betareg` for details.

`get_auxiliary_functions.glm` *Function to compute/extract auxiliary functions from objects of class* `glm/enriched_glm`

Description

Function to compute/extract auxiliary functions from objects of class `glm/enriched_glm`

Usage

```
## S3 method for class 'glm'
get_auxiliary_functions(object, ...)
```

Arguments

`object`	an object of class `glm` or`enriched_glm`
`...`	currently not used

Details

See `enrich.glm` for details.

get_auxiliary_functions.lm

Function to compute/extract auxiliary functions from objects of class `lm`/`enriched_lm`

Description

Function to compute/extract auxiliary functions from objects of class `lm`/`enriched_lm`

Usage

```
## S3 method for class 'lm'
get_auxiliary_functions(object, ...)
```

Arguments

`object`	an object of class `lm` or`enriched_lm`
`...`	currently not used

Details

See `enrich.lm` for details.

get_bias_function

Generic method for extracting or computing a function that returns the bias for the parameters in modelling objects

Description

Generic method for extracting or computing a function that returns the bias for the parameters in modelling objects

Usage

```
get_bias_function(object, ...)
```

Arguments

`object`	the object to be enriched or the enriched object
`...`	currently not used

get_bias_function.betareg *Function to compute/extract a function that returns the first term in the expansion of the bias of the MLE for the parameters of an object of class* betareg/enriched_betareg

Description

Function to compute/extract a function that returns the first term in the expansion of the bias of the MLE for the parameters of an object of class betareg/enriched_betareg

Usage

```
## S3 method for class 'betareg'
get_bias_function(object, ...)
```

Arguments

object	an object of class betareg orenriched_betareg
...	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the first-order bias is evacuated. If missing then the maximum likelihood estimates are used

get_bias_function.glm *Function to compute/extract a function that returns the first term in the expansion of the bias of the MLE for the parameters of an object of class* glm/enriched_glm

Description

Function to compute/extract a function that returns the first term in the expansion of the bias of the MLE for the parameters of an object of class glm/enriched_glm

Usage

```
## S3 method for class 'glm'
get_bias_function(object, ...)
```

Arguments

object	an object of class glm orenriched_glm
...	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the first-order bias is evacuated. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the first-order bias is evaluated. If missing then the maximum likelihood estimate is used

---

`get_bias_function.lm` *Function to compute/extract a function that returns the first term in the expansion of the bias of the MLE for the parameters of an object of class* `lm`/`enriched_lm`

Description

Function to compute/extract a function that returns the first term in the expansion of the bias of the MLE for the parameters of an object of class `lm`/`enriched_lm`

Usage

```
## S3 method for class 'lm'
get_bias_function(object, ...)
```

Arguments

`object`	an object of class `lm` or`enriched_lm`
`...`	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the first-order bias is evacuated. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the first-order bias is evaluated. If missing then the maximum likelihood estimate is used

`get_dmodel_function` *Generic method for extracting or computing a dmodel function for modelling objects*

Description

Generic method for extracting or computing a dmodel function for modelling objects

Usage

```
get_dmodel_function(object, ...)
```

Arguments

`object`	the object to be enriched or the enriched object
`...`	currently not used

`get_dmodel_function.glm` *Function to compute/extract a* `dmodel` *function*

Description

Function to compute/extract a `dmodel` function

Usage

```
## S3 method for class 'glm'
get_dmodel_function(object, ...)
```

Arguments

`object`	an object of class `glm` or`enriched_glm`
`...`	currently not used

Details

The computed/extracted function has arguments

data a data frame with observations at which to compute densities. If missing then densities are computed at the model frame extracted from the object (see `glm`)

coefficients the regression coefficients at which the densities are computed. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the densities function is computed. If missing then the maximum likelihood estimate is used

log logical; if `TRUE`, the logarithm of the density is returned

get_enrichment_options *Generic method for getting available options for the enrichment of objects*

Description

Generic method for getting available options for the enrichment of objects

Usage

```
get_enrichment_options(object, option, all_options)
```

Arguments

object	the object to be enriched
option	a character vector listing the options for enriching the object
all_options	if TRUE then output a data frame with the available enrichment options, their descriptions, the names of the components that each option results in, and the names of the corresponding compute functions.

Details

A check is being made whether the requested option is available. No check is being made on whether the functions that produce the components exist.

Value

if all_options = TRUE then an object of class enrichment_options is returned, otherwise if option is specified the output is a character vector with the names of the functions that compute the enrichment components

See Also

enrich.glm, enrich.family, enrich.link-glm, enrich.betareg

`get_enrichment_options.betareg` *Available options for the enrichment objects of class betareg*

Description

Available options for the enrichment objects of class betareg

Usage

```
## S3 method for class 'betareg'
get_enrichment_options(object, option,
  all_options = missing(option))
```

Arguments

`object`	the object to be enriched
`option`	a character vector listing the options for enriching the object
`all_options`	if `TRUE` then output a data frame with the available enrichment options, their descriptions, the names of the components that each option results in, and the names of the corresponding `compute_*` functions.

Details

A check is being made whether the requested option is available. No check is being made on whether the functions that produce the components exist.

Value

an object of class `enrichment_options`

Examples

```
## Not run:
get_enrichment_options.betareg(option = "all")
get_enrichment_options.betareg(all_options = TRUE)

## End(Not run)
```

`get_enrichment_options.family` *Available options for the enrichment objects of class family*

Description

Available options for the enrichment objects of class family

Usage

```
## S3 method for class 'family'
get_enrichment_options(object, option,
  all_options = missing(option))
```

Arguments

`object`	the object to be enriched
`option`	a character vector listing the options for enriching the object
`all_options`	if `TRUE` then output a data frame with the available enrichment options, their descriptions, the names of the components that each option results in, and the names of the corresponding `compute_*` functions.

Details

A check is being made whether the requested option is available. No check is being made on whether the functions that produce the components exist.

Value

an object of class `enrichment_options`

Examples

```
## Not run:
get_enrichment_options.family(option = "all")
get_enrichment_options.family(all_options = TRUE)

## End(Not run)
```

`get_enrichment_options.glm` *Available options for the enrichment objects of class* `glm`

Description

Available options for the enrichment objects of class `glm`

Usage

```
## S3 method for class 'glm'
get_enrichment_options(object, option,
  all_options = missing(option))
```

Arguments

`object`	the object to be enriched
`option`	a character vector listing the options for enriching the object
`all_options`	if `TRUE` then output a data frame with the available enrichment options, their descriptions, the names of the components that each option results in, and the names of the corresponding `compute_*` functions.

Details

A check is being made whether the requested option is available. No check is being made on whether the functions that produce the components exist.

Value

an object of class `enrichment_options`

Examples

```
## Not run:
get_enrichment_options.glm(option = "all")
get_enrichment_options.glm(all_options = TRUE)

## End(Not run)
```

`get_enrichment_options.link-glm` *Available options for the enrichment objects of class link-glm*

Description

Available options for the enrichment objects of class link-glm

Usage

```
## S3 method for class '`link-glm`'
get_enrichment_options(object, option,
  all_options = missing(option))
```

Arguments

`object`	the object to be enriched
`option`	a character vector listing the options for enriching the object
`all_options`	if `TRUE` then output a data frame with the available enrichment options, their descriptions, the names of the components that each option results in, and the names of the corresponding `compute_*` functions.

Details

A check is being made whether the requested option is available. No check is being made on whether the functions that produce the components exist.

Value

an object of class `enrichment_options`

Examples

```
## Not run:
`get_enrichment_options.link-glm`(option = "all")
`get_enrichment_options.link-glm`(all_options = TRUE)

## End(Not run)
```

`get_enrichment_options.lm` *Available options for the enrichment objects of class* `lm`

Description

Available options for the enrichment objects of class `lm`

Usage

```
## S3 method for class 'lm'
get_enrichment_options(object, option,
  all_options = missing(option))
```

Arguments

`object`	the object to be enriched
`option`	a character vector listing the options for enriching the object
`all_options`	if `TRUE` then output a data frame with the available enrichment options, their descriptions, the names of the components that each option results in, and the names of the corresponding `compute_*` functions.

Details

A check is being made whether the requested option is available. No check is being made on whether the functions that produce the components exist.

Value

an object of class `enrichment_options`

Examples

```
## Not run:
get_enrichment_options.lm(option = "all")
get_enrichment_options.lm(all_options = TRUE)

## End(Not run)
```

get_information_function

Generic method for extracting or computing a function that returns the information matrix for modelling objects

Description

Generic method for extracting or computing a function that returns the information matrix for modelling objects

Usage

```
get_information_function(object, ...)
```

Arguments

object	the object to be enriched or the enriched object
...	currently not used

get_information_function.betareg

Function to compute/extract a function that returns the information matrix for an object of class betareg/enriched_betareg

Description

Function to compute/extract a function that returns the information matrix for an object of class betareg/enriched_betareg

Usage

```
## S3 method for class 'betareg'
get_information_function(object, ...)
```

Arguments

object	an object of class betareg orenriched_betareg
...	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the information matrix is evaluated. If missing then the maximum likelihood estimates are used

type should the function return th 'expected' or 'observed' information? Default is `expected`

QR Currently not used

CHOL If `TRUE`, then the Cholesky decomposition of the information matrix at the coefficients is returned

---

get_information_function.glm

Function to compute/extract a function that returns the information matrix for an object of class `glm`/`enriched_glm`

Description

Function to compute/extract a function that returns the information matrix for an object of class `glm`/`enriched_glm`

Usage

```
## S3 method for class 'glm'
get_information_function(object, ...)
```

Arguments

`object`	an object of class `glm` or`enriched_glm`
`...`	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the information matrix is evaluated. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the information matrix is evaluated. If missing then the maximum likelihood estimate is used

type should the function return th 'expected' or 'observed' information? Default is `expected`

QR If `TRUE`, then the QR decomposition of

$$W^{1/2}X$$

is returned, where

$$W$$

is a diagonal matrix with the working weights (`object$weights`) and

$$X$$

is the model matrix.

CHOL If `TRUE`, then the Cholesky decomposition of the information matrix at the coefficients is returned

---

`get_information_function.lm`

Function to compute/extract a function that returns the information matrix for an object of class `lm`/`enriched_lm`

Description

Function to compute/extract a function that returns the information matrix for an object of class `lm`/`enriched_lm`

Usage

```
## S3 method for class 'lm'
get_information_function(object, ...)
```

Arguments

`object`	an object of class `lm` or`enriched_lm`
`...`	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the information matrix is evaluated. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the information matrix is evaluated. If missing then the maximum likelihood estimate is used

type should the function return th 'expected' or 'observed' information? Default is `expected`

QR If `TRUE`, then the QR decomposition of

$$W^{1/2}X$$

is returned, where

$$W$$

is a diagonal matrix with the working weights (`object$weights`) and

$$X$$

is the model matrix.

CHOL If `TRUE`, then the Cholesky decomposition of the information matrix at the coefficients is returned

`get_pmodel_function` *Generic method for extracting or computing a pmodel function for modelling objects*

Description

Generic method for extracting or computing a pmodel function for modelling objects

Usage

```
get_pmodel_function(object, ...)
```

Arguments

`object`	the object to be enriched or the enriched object
`...`	currently not used

`get_pmodel_function.glm` *Function to compute/extract a* `pmodel` *function*

Description

Function to compute/extract a `pmodel` function

Usage

```
## S3 method for class 'glm'
get_pmodel_function(object, ...)
```

Arguments

`object`	an object of class `glm` or`enriched_glm`
`...`	currently not used

Details

The computed/extracted function has arguments

data a data frame with observations at which to compute the distribution function. If missing then probabilities are computed at the model frame extracted from the object (see `glm`)

coefficients the regression coefficients at which the distribution function are computed. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the distribution function is computed. If missing then the maximum likelihood estimate is used

log.p logical; if `TRUE`, the logarithm of the distribution function is returned

lower.tail logical; if `TRUE` (default), probabilities are P[X <= x] otherwise, P[X > x]

get_qmodel_function *Generic method for extracting or computing a qmodel function for modelling objects*

Description

Generic method for extracting or computing a qmodel function for modelling objects

Usage

```
get_qmodel_function(object, ...)
```

Arguments

object	the object to be enriched or the enriched object
...	currently not used

get_qmodel_function.glm *Function to compute/extract a* qmodel *function*

Description

Function to compute/extract a qmodel function

Usage

```
## S3 method for class 'glm'
get_qmodel_function(object, ...)
```

Arguments

object	an object of class glm orenriched_glm
...	currently not used

Details

The computed/extracted function has arguments

p a vector of probabilities with length(p) equal to nrow(data) at which to evaluate quantiles

data a data frame with observations at which to compute the quantiles. If missing then quantiles are computed at the model frame extracted from the object (see glm)

coefficients the regression coefficients at which the quantiles are computed. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the quantiles are computed. If missing then the maximum likelihood estimate is used

log.p logical; if TRUE, the logarithm of the probabilities is used

lower.tail logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x]

`get_score_function` *Generic method for extracting or computing a function that returns the scores for modelling objects*

Description

Generic method for extracting or computing a function that returns the scores for modelling objects

Usage

```
get_score_function(object, ...)
```

Arguments

`object`	the object to be enriched or the enriched object
`...`	currently not used

`get_score_function.betareg` *Function to compute/extract a function that returns the scores (derivatives of the log-likelihood) for an object of class* `betareg/enriched_betareg`

Description

Function to compute/extract a function that returns the scores (derivatives of the log-likelihood) for an object of class `betareg/enriched_betareg`

Usage

```
## S3 method for class 'betareg'
get_score_function(object, ...)
```

Arguments

`object`	an object of class `betareg` or`enriched_betareg`
`...`	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the scores are computed. If missing then the maximum likelihood estimates are used

---

`get_score_function.glm`

Function to compute/extract a function that returns the scores (derivatives of the log-likelihood) for an object of class `glm`/`enriched_glm`

Description

Function to compute/extract a function that returns the scores (derivatives of the log-likelihood) for an object of class `glm`/`enriched_glm`

Usage

```
## S3 method for class 'glm'
get_score_function(object, ...)
```

Arguments

`object`	an object of class `glm` or`enriched_glm`
`...`	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the scores are computed. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the score function is evaluated. If missing then the maximum likelihood estimate is used

get_score_function.lm *Function to compute/extract a function that returns the scores (derivatives of the log-likelihood) for an object of class* lm/enriched_lm

Description

Function to compute/extract a function that returns the scores (derivatives of the log-likelihood) for an object of class lm/enriched_lm

Usage

```
## S3 method for class 'lm'
get_score_function(object, ...)
```

Arguments

object	an object of class lm orenriched_lm
...	currently not used

Details

The computed/extracted function has arguments

coefficients the regression coefficients at which the scores are computed. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the score function is evaluated. If missing then the maximum likelihood estimate is used

get_simulate_function *Generic method for extracting or computing a simulate function for modelling objects*

Description

Generic method for extracting or computing a simulate function for modelling objects

Usage

```
get_simulate_function(object, ...)
```

Arguments

object	the object to be enriched or the enriched object
...	currently not used

get_simulate_function.betareg

Function to compute/extract a simulate function for response vectors from an object of class betareg/enriched_betareg

Description

Function to compute/extract a simulate function for response vectors from an object of class betareg/enriched_betareg

Usage

```
## S3 method for class 'betareg'
get_simulate_function(object, ...)
```

Arguments

object	an object of class betareg orenriched_betareg
...	currently not used

Details

The computed/extracted simulate function has arguments

coefficients the regression coefficients at which the response vectors are simulated. If missing then the maximum likelihood estimates are used

nsim number of response vectors to simulate. Defaults to 1

seed an object specifying if and how the random number generator should be initialized ('seeded'). It can be either NULL or an integer that will be used in a call to set.seed before simulating the response vectors. If set, the value is saved as the seed attribute of the returned value. The default, NULL will not change the random generator state, and return .Random.seed as the seed attribute, see Value

get_simulate_function.glm

Function to compute/extract a simulate function for response vectors from an object of class glm/enriched_glm

Description

Function to compute/extract a simulate function for response vectors from an object of class glm/enriched_glm

Usage

```
## S3 method for class 'glm'
get_simulate_function(object, ...)
```

Arguments

object	an object of class glm orenriched_glm
...	currently not used

Details

The computed/extracted simulate function has arguments

coefficients the regression coefficients at which the response vectors are simulated. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the response vectors are simulated. If missing then the maximum likelihood estimate is used

nsim number of response vectors to simulate. Defaults to 1

seed an object specifying if and how the random number generator should be initialized ('seeded'). It can be either NULL or an integer that will be used in a call to set.seed before simulating the response vectors. If set, the value is saved as the seed attribute of the returned value. The default, NULL will not change the random generator state, and return .Random.seed as the seed attribute, see Value

---

get_simulate_function.lm

Function to compute/extract a simulate function for response vectors from an object of class lm/enriched_lm

---

Description

Function to compute/extract a simulate function for response vectors from an object of class lm/enriched_lm

Usage

```
## S3 method for class 'lm'
get_simulate_function(object, ...)
```

Arguments

object	an object of class lm orenriched_lm
...	currently not used

Details

The computed/extracted simulate function has arguments

coefficients the regression coefficients at which the response vectors are simulated. If missing then the maximum likelihood estimates are used

dispersion the dispersion parameter at which the response vectors are simulated. If missing then the maximum likelihood estimate is used

nsim number of response vectors to simulate. Defaults to 1

seed an object specifying if and how the random number generator should be initialized ('seeded'). It can be either NULL or an integer that will be used in a call to set.seed before simulating the response vectors. If set, the value is saved as the seed attribute of the returned value. The default, NULL will not change the random generator state, and return .Random.seed as the seed attribute, see Value

Index