

Package ‘prediction’

July 23, 2025

Type Package

Title Tidy, Type-Safe 'prediction()' Methods

Description A one-function package containing prediction(), a type-safe alternative to predict() that always returns a data frame. The summary() method provides a data frame with average predictions, possibly over counterfactual versions of the data (à la the margins command in 'Stata'). Marginal effect estimation is provided by the related package, 'margins' <<https://cran.r-project.org/package=margins>>. The package currently supports common model types (e.g., lm, glm) from the 'stats' package, as well as numerous other model classes from other add-on packages. See the README file or main package documentation page for a complete listing.

License MIT + file LICENSE

Version 0.3.18

URL <https://github.com/bbolker/prediction>

BugReports <https://github.com/bbolker/prediction/issues>

Depends R (>= 3.5.0)

Imports utils, stats, data.table

Suggests datasets, methods, testthat

Enhances AER, aod, betareg, biglm, brglm, caret, crch, e1071, earth, ff, gam (>= 1.15), gee, glmnet, glmx, kernlab, lme4, MASS, mclogit, mda, mlogit, MNP, nlme, nnet, ordinal, plm, pscl, quantreg, rpart, sampleSelection, speedglm, survey (>= 3.31-5), survival, truncreg, VGAM

ByteCompile true

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no

Author Thomas J. Leeper [aut] (ORCID: <<https://orcid.org/0000-0003-4097-6326>>),
Carl Ganz [ctb],
Vincent Arel-Bundock [ctb] (ORCID:
<<https://orcid.org/0000-0003-2042-7063>>),
Ben Bolker [ctb, cre] (ORCID: <<https://orcid.org/0000-0002-2127-0443>>)

Maintainer Ben Bolker <bolker@mcmaster.ca>
Repository CRAN
Date/Publication 2024-06-11 15:20:02 UTC

Contents

prediction-package	2
build_datalist	18
find_data	19
margex	21
mean_or_mode	23
seq_range	24
Index	26

prediction-package	<i>Extract Predictions from a Model Object</i>
--------------------	--

Description

Extract predicted values via [predict](#) from a model object, conditional on data, and return a data frame.

Usage

```
prediction(model, ...)

## Default S3 method:
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = "response",
  vcov = stats::vcov(model),
  calculate_se = TRUE,
  ...
)

## S3 method for class 'Arima'
prediction(model, calculate_se = TRUE, ...)

## S3 method for class 'ar'
prediction(model, data, at = NULL, calculate_se = TRUE, ...)

## S3 method for class 'arima0'
prediction(model, data, at = NULL, calculate_se = TRUE, ...)
```

```
## S3 method for class 'betareg'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link", "precision", "variance", "quantile"),
  calculate_se = FALSE,
  ...
)

## S3 method for class 'biglm'
prediction(model, data = NULL, calculate_se = FALSE, ...)

## S3 method for class 'bigglm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = "response",
  calculate_se = TRUE,
  ...
)

## S3 method for class 'biglm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = "response",
  calculate_se = TRUE,
  ...
)

## S3 method for class 'bruto'
prediction(
  model,
  data = NULL,
  at = NULL,
  type = "fitted",
  calculate_se = FALSE,
  ...
)

## S3 method for class 'clm'
prediction(
  model,
  data = find_data(model, parent.frame()),
```

```
    at = NULL,
    type = NULL,
    calculate_se = TRUE,
    category,
    ...
)

## S3 method for class 'coxph'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("risk", "expected", "lp"),
  calculate_se = TRUE,
  ...
)

## S3 method for class 'crch'
prediction(
  model,
  data = find_data(model),
  at = NULL,
  type = c("response", "location", "scale", "quantile"),
  calculate_se = FALSE,
  ...
)

## S3 method for class 'earth'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = TRUE,
  category,
  ...
)

## S3 method for class 'fda'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = NULL,
  calculate_se = FALSE,
  category,
  ...
)
```

```
## S3 method for class 'Gam'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link", "terms"),
  calculate_se = TRUE,
  ...
)

## S3 method for class 'gausspr'
prediction(
  model,
  data,
  at = NULL,
  type = NULL,
  calculate_se = TRUE,
  category,
  ...
)

## S3 method for class 'gee'
prediction(model, calculate_se = FALSE, ...)

## S3 method for class 'glimML'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = TRUE,
  ...
)

## S3 method for class 'glimQL'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = TRUE,
  ...
)

## S3 method for class 'glm'
prediction(
  model,
```

```

    data = find_data(model, parent.frame()),
    at = NULL,
    type = c("response", "link"),
    vcov = stats::vcov(model),
    calculate_se = TRUE,
    ...
)

## S3 method for class 'glmnet'
prediction(
  model,
  data,
  lambda = model[["lambda"]][1L],
  at = NULL,
  type = c("response", "link"),
  calculate_se = FALSE,
  ...
)

## S3 method for class 'glm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = FALSE,
  ...
)

## S3 method for class 'glsl'
prediction(
  model,
  data = find_data(model),
  at = NULL,
  calculate_se = FALSE,
  ...
)

## S3 method for class 'hetglm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link", "scale"),
  calculate_se = FALSE,
  ...
)

```

```
## S3 method for class 'hurdle'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "count", "prob", "zero"),
  calculate_se = FALSE,
  ...
)

## S3 method for class 'hxlr'
prediction(
  model,
  data = find_data(model),
  at = NULL,
  type = c("class", "probability", "cumprob", "location", "scale"),
  calculate_se = FALSE,
  ...
)

## S3 method for class 'ivreg'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  calculate_se = FALSE,
  ...
)

## S3 method for class 'knnreg'
prediction(model, data, at = NULL, calculate_se = FALSE, ...)

## S3 method for class 'kqr'
prediction(model, data, at = NULL, calculate_se = FALSE, ...)

## S3 method for class 'ksvm'
prediction(
  model,
  data,
  at = NULL,
  type = NULL,
  calculate_se = TRUE,
  category,
  ...
)

## S3 method for class 'lda'
prediction(
```

```
    model,
    data = find_data(model, parent.frame()),
    at = NULL,
    calculate_se = FALSE,
    category,
    ...
)

## S3 method for class 'lm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = "response",
  vcov = stats::vcov(model),
  calculate_se = TRUE,
  ...
)

## S3 method for class 'lme'
prediction(
  model,
  data = find_data(model),
  at = NULL,
  calculate_se = FALSE,
  ...
)

## S3 method for class 'loess'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = "response",
  calculate_se = TRUE,
  ...
)

## S3 method for class 'lqs'
prediction(
  model,
  data = find_data(model),
  at = NULL,
  calculate_se = FALSE,
  ...
)

## S3 method for class 'mars'
```



```
prediction(  
  model,  
  data = NULL,  
  at = NULL,  
  type = "fitted",  
  calculate_se = FALSE,  
  ...  
)  
  
## S3 method for class 'mca'  
prediction(  
  model,  
  data = find_data(model),  
  at = NULL,  
  calculate_se = FALSE,  
  ...  
)  
  
## S3 method for class 'mclogit'  
prediction(  
  model,  
  data = find_data(model, parent.frame()),  
  at = NULL,  
  type = "response",  
  vcov = stats::vcov(model),  
  calculate_se = TRUE,  
  ...  
)  
  
## S3 method for class 'merMod'  
prediction(  
  model,  
  data = find_data(model),  
  at = NULL,  
  type = c("response", "link"),  
  re.form = NULL,  
  calculate_se = FALSE,  
  ...  
)  
  
## S3 method for class 'mlogit'  
prediction(  
  model,  
  data = find_data(model, parent.frame()),  
  at = NULL,  
  calculate_se = FALSE,  
  category,  
  ...  
)
```

```
)

## S3 method for class 'mnlogit'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  calculate_se = FALSE,
  category,
  ...
)

## S3 method for class 'mnp'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = NULL,
  calculate_se = FALSE,
  category,
  ...
)

## S3 method for class 'multinom'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = NULL,
  calculate_se = FALSE,
  category,
  ...
)

## S3 method for class 'naiveBayes'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = NULL,
  calculate_se = FALSE,
  category,
  ...
)

## S3 method for class 'nls'
prediction(
  model,
```

```
    data = find_data(model, parent.frame()),
    at = NULL,
    calculate_se = FALSE,
    ...
)

## S3 method for class 'nnet'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = NULL,
  calculate_se = FALSE,
  category,
  ...
)

## S3 method for class 'plm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  calculate_se = FALSE,
  ...
)

## S3 method for class 'polr'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = NULL,
  calculate_se = FALSE,
  category,
  ...
)

## S3 method for class 'polyreg'
prediction(
  model,
  data = NULL,
  at = NULL,
  type = "fitted",
  calculate_se = FALSE,
  ...
)

## S3 method for class 'ppr'
```

```
prediction(  
  model,  
  data = find_data(model, parent.frame()),  
  at = NULL,  
  calculate_se = FALSE,  
  ...  
)  
  
## S3 method for class 'princomp'  
prediction(  
  model,  
  data = find_data(model, parent.frame()),  
  at = NULL,  
  calculate_se = FALSE,  
  ...  
)  
  
## S3 method for class 'qda'  
prediction(  
  model,  
  data = find_data(model, parent.frame()),  
  at = NULL,  
  calculate_se = FALSE,  
  category,  
  ...  
)  
  
## S3 method for class 'rlm'  
prediction(  
  model,  
  data = find_data(model, parent.frame()),  
  at = NULL,  
  type = "response",  
  vcov = stats::vcov(model),  
  calculate_se = TRUE,  
  ...  
)  
  
## S3 method for class 'rpart'  
prediction(  
  model,  
  data = find_data(model, parent.frame()),  
  at = NULL,  
  type = NULL,  
  calculate_se = FALSE,  
  category,  
  ...  
)
```

```
## S3 method for class 'rq'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  calculate_se = TRUE,
  ...
)

## S3 method for class 'selection'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = "response",
  calculate_se = FALSE,
  ...
)

## S3 method for class 'speedglm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = FALSE,
  ...
)

## S3 method for class 'speedlm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  calculate_se = FALSE,
  ...
)

## S3 method for class 'survreg'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "lp", "quantile", "uquantile"),
  calculate_se = TRUE,
  ...
)
```

```
## S3 method for class 'svm'
prediction(model, data = NULL, at = NULL, calculate_se = TRUE, category, ...)

## S3 method for class 'svyglm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = TRUE,
  ...
)

## S3 method for class 'train'
prediction(
  model,
  data = find_data(model),
  at = NULL,
  type = c("raw", "prob"),
  ...
)

## S3 method for class 'tree'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = NULL,
  calculate_se = FALSE,
  category,
  ...
)

## S3 method for class 'truncreg'
prediction(model, data, at = NULL, calculate_se = FALSE, ...)

## S3 method for class 'vgam'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = FALSE,
  category,
  ...
)
```

```
## S3 method for class 'vglm'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "link"),
  calculate_se = TRUE,
  category,
  ...
)

## S3 method for class 'zeroinfl'
prediction(
  model,
  data = find_data(model, parent.frame()),
  at = NULL,
  type = c("response", "count", "prob", "zero"),
  calculate_se = FALSE,
  ...
)

prediction_summary(model, ..., level = 0.95)
```

Arguments

<code>model</code>	A model object, perhaps returned by <code>lm</code> or <code>glm</code> .
<code>...</code>	Additional arguments passed to <code>predict</code> methods.
<code>data</code>	A <code>data.frame</code> over which to calculate marginal effects. If missing, <code>find_data</code> is used to specify the data frame.
<code>at</code>	A list of one or more named vectors, specifically values at which to calculate the predictions. These are used to modify the value of <code>data</code> (see <code>build_datalist</code> for details on use).
<code>type</code>	A character string indicating the type of marginal effects to estimate. Mostly relevant for non-linear models, where the reasonable options are “response” (the default) or “link” (i.e., on the scale of the linear predictor in a GLM). For models of class “polr” (from <code>polr</code>), possible values are “class” or “probs”; both are returned.
<code>vcov</code>	A matrix containing the variance-covariance matrix for estimated model coefficients, or a function to perform the estimation with <code>model</code> as its only argument.
<code>calculate_se</code>	A logical indicating whether to calculate standard errors for observation-specific predictions and average predictions (if possible). The output will always contain a “calculate_se” column regardless of this value; this only controls the calculation of standard errors. Setting it to <code>FALSE</code> may improve speed.
<code>category</code>	For multi-level or multi-category outcome models (e.g., ordered probit, multinomial logit, etc.), a value specifying which of the outcome levels should be used for the “fitted” column. If missing, some default is chosen automatically.

<code>lambda</code>	For models of class “glmnet”, a value of the penalty parameter at which predictions are required.
<code>re.form</code>	An argument passed forward to predict.merMod .
<code>level</code>	A numeric value specifying the confidence level for calculating p-values and confidence intervals.

Details

This function is simply a wrapper around [predict](#) that returns a data frame containing the value of data and the predicted values with respect to all variables specified in data.

Methods are currently implemented for the following object classes:

- “lm”, see [lm](#)
- “glm”, see [glm](#), [glm.nb](#), [glmX](#), [hetglm](#), [brglm](#)
- “ar”, see [ar](#)
- “Arima”, see [arima](#)
- “arima0”, see [arima0](#)
- “bigglm”, see [bigglm](#)
- “betareg”, see [betareg](#)
- “bruto”, see [bruto](#)
- “clm”, see [clm](#)
- “coxph”, see [coxph](#)
- “crch”, see [crch](#)
- “earth”, see [earth](#)
- “fda”, see [fda](#)
- “Gam”, see [gam](#)
- “gausspr”, see [gausspr](#)
- “gee”, see [gee](#)
- “glmnet”, see [glmnet](#)
- “gls”, see [gls](#)
- “glimML”, see [betabin](#), [negbin](#)
- “glimQL”, see [quasibin](#), [quasipois](#)
- “hurdle”, see [hurdle](#)
- “hxlr”, see [hxlr](#)
- “ivreg”, see [ivreg](#)
- “knnreg”, see [knnreg](#)
- “kqr”, see [kqr](#)
- “ksvm”, see [ksvm](#)
- “lda”, see [lda](#)
- “lme”, see [lme](#)

- “loess”, see [loess](#)
- “lqs”, see [lqs](#)
- “mars”, see [mars](#)
- “mca”, see [mca](#)
- “mclogit”, see [mclogit](#)
- “mda”, see [mda](#)
- “merMod”, see [lmer](#), [glmer](#)
- “mnp”, see [mnp](#)
- “naiveBayes”, see [naiveBayes](#)
- “nlme”, see [nlme](#)
- “nls”, see [nls](#)
- “nnet”, see [nnet](#)
- “plm”, see [plm](#)
- “polr”, see [polr](#)
- “polyreg”, see [polyreg](#)
- “ppr”, see [ppr](#)
- “princomp”, see [princomp](#)
- “qda”, see [qda](#)
- “rlm”, see [rlm](#)
- “rpart”, see [rpart](#)
- “rq”, see [rq](#)
- “selection”, see [selection](#)
- “speedglm”, see [speedglm](#)
- “speedlm”, see [speedlm](#)
- “survreg”, see [survreg](#)
- “svm”, see [svm](#)
- “svyglm”, see [svyglm](#)
- “tobit”, see [tobit](#)
- “train”, see [train](#)
- “truncreg”, see [truncreg](#)
- “zeroinfl”, see [zeroinfl](#)

Where implemented, `prediction` also returns average predictions (and the variances thereof). Variances are implemented using the delta method, as described by Xu and Long 2005 [doi:10.1177/1536867X0500500405](https://doi.org/10.1177/1536867X0500500405).

Value

A data frame with class “prediction” that has a number of rows equal to number of rows in data, or a multiple thereof, if `!is.null(at)`. The return value contains data (possibly modified by `at` using `build_datalist`), plus a column containing fitted/predicted values (“fitted”) and a column containing the standard errors thereof (“calculate_se”). Additional columns may be reported depending on the object class. The data frame also carries attributes used by `print` and `summary`, which will be lost during subsetting.

See Also

`find_data`, `build_datalist`, `mean_or_mode`, `seq_range`

Examples

```
require("datasets")
x <- lm(Petal.Width ~ Sepal.Length * Sepal.Width * Species, data = iris)
# prediction for every case
prediction(x)

# prediction for first case
prediction(x, iris[1,])

# basic use of 'at' argument
summary(prediction(x, at = list(Species = c("setosa", "virginica"))))

# basic use of 'at' argument
prediction(x, at = list(Sepal.Length = seq_range(iris$Sepal.Length, 5)))

# prediction at means/modes of input variables
prediction(x, at = lapply(iris, mean_or_mode))

# prediction with multi-category outcome
## Not run:
library("mlogit")
data("Fishing", package = "mlogit")
Fish <- mlogit.data(Fishing, varying = c(2:9), shape = "wide", choice = "mode")
mod <- mlogit(mode ~ price + catch, data = Fish)
prediction(mod)
prediction(mod, category = 3)

## End(Not run)
```

build_datalist

Build list of data.frames

Description

Construct a list of data.frames based upon an input data.frame and a list of one or more `at` values

Usage

```
build_datalist(data, at = NULL, as.data.frame = FALSE, ...)
```

Arguments

<code>data</code>	A data.frame containing the original data.
<code>at</code>	A list of one or more named vectors of values, which will be used to specify values of variables in data. All possible combinations are generated. Alternatively, this can be a data frame of combination levels if only a subset of combinations are desired. See examples.
<code>as.data.frame</code>	A logical indicating whether to return a single stacked data frame rather than a list of data frames
<code>...</code>	Ignored.

Value

A list of data.frames, unless `as.data.frame = TRUE` in which case a single, stacked data frame is returned.

Author(s)

Thomas J. Leeper

See Also

[find_data](#), [mean_or_mode](#), [seq_range](#)

Examples

```
# basic examples
require("datasets")
build_datalist(head(mtcars), at = list(cyl = c(4, 6)))

str(build_datalist(head(mtcars), at = list(cyl = c(4,6), wt = c(2.75,3,3.25))), 1)

str(build_datalist(head(mtcars), at = data.frame(cyl = c(4,4), wt = c(2.75,3))))
```

find_data

Extract data from a model object

Description

Attempt to reconstruct the data used to create a model object

Usage

```
find_data(model, ...)

## Default S3 method:
find_data(model, env = parent.frame(), ...)

## S3 method for class 'data.frame'
find_data(model, ...)

## S3 method for class 'crch'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'glimML'
find_data(model, ...)

## S3 method for class 'glimQL'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'glm'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'hxlr'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'lm'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'mca'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'merMod'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'svyglm'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'train'
find_data(model, ...)

## S3 method for class 'vgam'
find_data(model, env = parent.frame(), ...)

## S3 method for class 'vglm'
find_data(model, env = parent.frame(), ...)
```

Arguments

model	The model object.
-------	-------------------

... Additional arguments passed to methods.
 env An environment in which to look for the data argument to the modelling call.

Details

This is a convenience function and, as such, carries no guarantees. To behave well, it typically requires that a model object be specified using a formula interface and an explicit data argument. Models that can be specified using variables from the `.GlobalEnv` or with a non-formula interface (e.g., a matrix of data) will tend to generate errors. `find_data` is an S3 generic so it is possible to expand it with new methods.

Value

A data frame containing the original data used in a modelling call, modified according to the original model's 'subset' and 'na.action' arguments, if appropriate.

See Also

[prediction](#), [build_datalist](#), [mean_or_mode](#), [seq_range](#)

Examples

```
require("datasets")
x <- lm(mpg ~ cyl * hp + wt, data = head(mtcars))
find_data(x)
```

margex

Artificial data for margins, copied from Stata

Description

The dataset is identical to the one provided by Stata and available from `webuse : webuse("margex")` with categorical variables explicitly encoded as factors.

Usage

```
data("margex")
```

Format

A data frame with 3000 observations on the following 11 variables.

'y' A numeric vector
 'outcome' A binary numeric vector with values (0,1)
 'sex' A factor with two levels
 'group' A factor with three levels

‘age’ A numeric vector
 ‘distance’ A numeric vector
 ‘ycn’ A numeric vector
 ‘yc’ A numeric vector
 ‘treatment’ A factor with two levels
 ‘agegroup’ A factor with three levels
 ‘arm’ A factor with three levels

Source

<https://www.stata-press.com/data/r14/margex.dta>

See Also

[prediction](#)

Examples

```

# Examples from Stata's help files
# Also available from: webuse::webuse("margex")
data("margex")

# A simple case after regress
# . regress y i.sex i.group
# . margins sex
m1 <- lm(y ~ factor(sex) + factor(group), data = margex)
prediction(m1, at = list(sex = c("male", "female")))

# A simple case after logistic
# . logistic outcome i.sex i.group
# . margins sex
m2 <- glm(outcome ~ sex + group, binomial(), data = margex)
prediction(m2, at = list(sex = c("male", "female")))

# Average response versus response at average
# . margins sex
prediction(m2, at = list(sex = c("male", "female")))
# . margins sex, atmeans
## TODO

# Multiple margins from one margins command
# . margins sex group
prediction(m2, at = list(sex = c("male", "female")))
prediction(m2, at = list(group = c("1", "2", "3")))

# Margins with interaction terms
# . logistic outcome i.sex i.group sex#group
# . margins sex group
m3 <- glm(outcome ~ sex * group, binomial(), data = margex)

```

```

prediction(m3, at = list(sex = c("male", "female")))
prediction(m3, at = list(group = c("1", "2", "3")))

# Margins with continuous variables
# . logistic outcome i.sex i.group sex#group age
# . margins sex group
m4 <- glm(outcome ~ sex * group + age, binomial(), data = margex)
prediction(m4, at = list(sex = c("male", "female")))
prediction(m4, at = list(group = c("1", "2", "3")))

# Margins of continuous variables
# . margins, at(age=40)
prediction(m4, at = list(age = 40))
# . margins, at(age=(30 35 40 45 50))
prediction(m4, at = list(age = c(30, 35, 40, 45, 50)))

# Margins of interactions
# . margins sex#group
prediction(m4, at = list(sex = c("male", "female"), group = c("1", "2", "3")))

```

mean_or_mode

Class-dependent variable aggregation

Description

Summarize a vector/variable into a single number, either a mean (median) for numeric vectors or the mode for categorical (character, factor, ordered, or logical) vectors. Useful for aggregation.

Usage

```

mean_or_mode(x)

## Default S3 method:
mean_or_mode(x)

## S3 method for class 'numeric'
mean_or_mode(x)

## S3 method for class 'data.frame'
mean_or_mode(x)

median_or_mode(x)

## Default S3 method:
median_or_mode(x)

## S3 method for class 'numeric'

```

```
median_or_mode(x)

## S3 method for class 'data.frame'
median_or_mode(x)
```

Arguments

x A vector.

Value

A numeric or factor vector of length 1.

See Also

[prediction](#), [build_datalist](#), [seq_range](#)

Examples

```
require("datasets")
# mean for numerics
mean_or_mode(iris)
mean_or_mode(iris[["Sepal.Length"]])
mean_or_mode(iris[["Species"]])

# median for numerics
median_or_mode(iris)
```

seq_range

Create a sequence over the range of a vector

Description

Define a sequence of evenly spaced values from the minimum to the maximum of a vector

Usage

```
seq_range(x, n = 2)
```

Arguments

x A numeric vector

n An integer specifying the length of sequence (i.e., number of points across the range of x)

Value

A vector of length n.

See Also

[mean_or_mode](#), [build_datalist](#)

Examples

```
identical(range(1:5), seq_range(1:5, n = 2))  
seq_range(1:5, n = 3)
```

Index

- * **datasets**
 - margex, [21](#)
- * **data**
 - build_datalist, [18](#)
- * **manip**
 - build_datalist, [18](#)
- * **models**
 - prediction-package, [2](#)
- ar, [16](#)
- arima, [16](#)
- arima0, [16](#)
- betabin, [16](#)
- betareg, [16](#)
- bigglm, [16](#)
- brglm, [16](#)
- bruto, [16](#)
- build_datalist, [15](#), [18](#), [18](#), [21](#), [24](#), [25](#)
- clm, [16](#)
- coxph, [16](#)
- crch, [16](#)
- earth, [16](#)
- fda, [16](#)
- find_data, [15](#), [18](#), [19](#), [19](#)
- gam, [16](#)
- gausspr, [16](#)
- gee, [16](#)
- glm, [15](#), [16](#)
- glm.nb, [16](#)
- glmer, [17](#)
- glmnet, [16](#)
- glmx, [16](#)
- glis, [16](#)
- hetglm, [16](#)
- hurdle, [16](#)
- hxr, [16](#)
- ivreg, [16](#)
- knnreg, [16](#)
- kqr, [16](#)
- ksvm, [16](#)
- lda, [16](#)
- lm, [15](#), [16](#)
- lme, [16](#)
- lmer, [17](#)
- loess, [17](#)
- lqs, [17](#)
- margex, [21](#)
- mars, [17](#)
- mca, [17](#)
- mclogit, [17](#)
- mda, [17](#)
- mean_or_mode, [18](#), [19](#), [21](#), [23](#), [25](#)
- median_or_mode (mean_or_mode), [23](#)
- mnf, [17](#)
- naiveBayes, [17](#)
- negbin, [16](#)
- nlme, [17](#)
- nls, [17](#)
- nnet, [17](#)
- plm, [17](#)
- polr, [15](#), [17](#)
- polyreg, [17](#)
- ppr, [17](#)
- predict, [2](#), [15](#), [16](#)
- predict.merMod, [16](#)
- prediction, [21](#), [22](#), [24](#)
- prediction (prediction-package), [2](#)
- prediction-package, [2](#)
- prediction.ar (prediction-package), [2](#)
- prediction.Arima (prediction-package), [2](#)

prediction.arima0 (prediction-package),
 2
 prediction.betareg
 (prediction-package), 2
 prediction.bigglm (prediction-package),
 2
 prediction.biglm (prediction-package), 2
 prediction.biglm (prediction-package), 2
 prediction.bruto (prediction-package), 2
 prediction.clm (prediction-package), 2
 prediction.coxph (prediction-package), 2
 prediction.crch (prediction-package), 2
 prediction.default
 (prediction-package), 2
 prediction.earth (prediction-package), 2
 prediction.fda (prediction-package), 2
 prediction.Gam (prediction-package), 2
 prediction.gausspr
 (prediction-package), 2
 prediction.gee (prediction-package), 2
 prediction.glimML (prediction-package),
 2
 prediction.glimQL (prediction-package),
 2
 prediction.glm (prediction-package), 2
 prediction.glmnet (prediction-package),
 2
 prediction.glmx (prediction-package), 2
 prediction.gls (prediction-package), 2
 prediction.hetglm (prediction-package),
 2
 prediction.hurdle (prediction-package),
 2
 prediction.hxlr (prediction-package), 2
 prediction.ivreg (prediction-package), 2
 prediction.knnreg (prediction-package),
 2
 prediction.kqr (prediction-package), 2
 prediction.ksvm (prediction-package), 2
 prediction.lda (prediction-package), 2
 prediction.lm (prediction-package), 2
 prediction.lme (prediction-package), 2
 prediction.loess (prediction-package), 2
 prediction.lqs (prediction-package), 2
 prediction.mars (prediction-package), 2
 prediction.mca (prediction-package), 2
 prediction.mclogit
 (prediction-package), 2
 prediction.merMod (prediction-package),
 2
 prediction.mlogit (prediction-package),
 2
 prediction.mnlogit
 (prediction-package), 2
 prediction.mnp (prediction-package), 2
 prediction.multinom
 (prediction-package), 2
 prediction.naiveBayes
 (prediction-package), 2
 prediction.nls (prediction-package), 2
 prediction.nnet (prediction-package), 2
 prediction.plm (prediction-package), 2
 prediction.polr (prediction-package), 2
 prediction.polyreg
 (prediction-package), 2
 prediction.ppr (prediction-package), 2
 prediction.princomp
 (prediction-package), 2
 prediction.qda (prediction-package), 2
 prediction.rlm (prediction-package), 2
 prediction.rpart (prediction-package), 2
 prediction.rq (prediction-package), 2
 prediction.selection
 (prediction-package), 2
 prediction.speedglm
 (prediction-package), 2
 prediction.speedlm
 (prediction-package), 2
 prediction.survreg
 (prediction-package), 2
 prediction.svm (prediction-package), 2
 prediction.svyglm (prediction-package),
 2
 prediction.train (prediction-package), 2
 prediction.tree (prediction-package), 2
 prediction.truncreg
 (prediction-package), 2
 prediction.vgam (prediction-package), 2
 prediction.vglm (prediction-package), 2
 prediction.zeroinfl
 (prediction-package), 2
 prediction_summary
 (prediction-package), 2
 princomp, 17
 qda, 17
 quasibin, 16

quasipois, [16](#)

rlm, [17](#)

rpart, [17](#)

rq, [17](#)

selection, [17](#)

seq_range, [18](#), [19](#), [21](#), [24](#), [24](#)

speedglm, [17](#)

speedlm, [17](#)

survreg, [17](#)

svm, [17](#)

svyglm, [17](#)

tobit, [17](#)

train, [17](#)

truncreg, [17](#)

zeroinfl, [17](#)