

# Package ‘nestedLogit’

March 6, 2026

**Title** Nested Dichotomy Logistic Regression Models

**Version** 0.4.0

**Date** 2026-03-01

**Description** Provides functions for specifying and fitting nested dichotomy logistic regression models for a multi-category response and methods for summarising and plotting those models. Nested dichotomies are statistically independent, and hence provide an additive decomposition of tests for the overall ‘polytomous’ response. When the dichotomies make sense substantively, this method can be a simpler alternative to the standard ‘multinomial’ logistic model which compares response categories to a reference level. See: J. Fox (2016), ‘Applied Regression Analysis and Generalized Linear Models’, 3rd Ed., ISBN 1452205663.

**License** GPL (>= 2)

**URL** <https://github.com/friendly/nestedLogit>,  
<https://friendly.github.io/nestedLogit/>

**BugReports** <https://github.com/friendly/nestedLogit/issues>

**Depends** R (>= 4.1.0)

**Imports** broom, car, dplyr, effects, graphics, grDevices, stats,  
stringr, tibble, scales

**Suggests** AER, carData, geomtextpath, ggplot2, ggeffects, here,  
insight, lobstr, knitr, nnet, parameters, performance,  
rmarkdown, see, spelling, testthat, tidyr, MASS, VGAM, mlogit,  
vcd

**VignetteBuilder** knitr, rmarkdown

**Encoding** UTF-8

**Language** en-US

**LazyData** TRUE

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** John Fox [aut] (ORCID: <<https://orcid.org/0000-0002-1196-8012>>),  
 Michael Friendly [aut, cre] (ORCID:  
 <<https://orcid.org/0000-0002-3237-0941>>),  
 Achim Zeileis [ctb] (ORCID: <<https://orcid.org/0000-0003-0918-3766>>)

**Maintainer** Michael Friendly <friendly@yorku.ca>

**Repository** CRAN

**Date/Publication** 2026-03-06 17:00:02 UTC

## Contents

as.data.frame.predictNestedLogit . . . . .	2
as.tree . . . . .	3
broomMethods . . . . .	5
Effect.nestedLogit . . . . .	6
gators . . . . .	7
GSS . . . . .	9
HealthInsurance . . . . .	10
models . . . . .	11
nestedHypotheses . . . . .	12
nestedLogit . . . . .	14
nestedMethods . . . . .	17
plot.nestedLogit . . . . .	19
predict.nestedLogit . . . . .	22
<b>Index</b>	<b>26</b>

---

as.data.frame.predictNestedLogit  
*Convert a Predicted Objects to a data.frame*

---

## Description

These functions provide simple ways to convert the results of `predict.nestedLogit` to a data frame in a consistent format for plotting and other actions.

## Usage

```
## S3 method for class 'predictNestedLogit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

## Arguments

x	a "predictNestedLogit" object
row.names	row.names for result (for conformity with generic; not currently used)
optional	logical. If TRUE, setting row names and converting column names (to syntactic names: see <code>make.names</code> is optional
...	other arguments (unused)

**Value**

- For `predict(..., model="nested")` (the default), returns a data frame containing the values of predictors along with the columns `response`, `p`, `se.p`, `logit`, `se.logit`.
- For `predict(..., model="dichotomies")`, returns a data frame containing the values of predictors along with the columns `response`, `logit`, and `se.logit`.

**Examples**

```
data("Women1f", package = "carData")
comparisons <- logits(work=dichotomy("not.work", c("parttime", "fulltime")),
                     full=dichotomy("parttime", "fulltime"))

wlf.nested <- nestedLogit(partic ~ hincome + children,
                        dichotomies = comparisons,
                        data=Women1f)
# get predicted values for a grid of `hincome` and `children`
new <- expand.grid(hincome=seq(0, 45, length=10),
                 children=c("absent", "present"))

pred.nested <- predict(wlf.nested, new)
plotdata <- as.data.frame(pred.nested)
str(plotdata)

# Predicted logit values for the dichotomies
pred.dichot <- predict(wlf.nested, new, model = "dichotomies")
plotlogit <- as.data.frame(pred.dichot)
str(plotlogit)
```

---

as.tree

*Display the Tree Structure of Nested Dichotomies*


---

**Description**

Display the nested structure of a "dichotomies" or "continuationDichotomies" object as a 2-D ASCII tree diagram showing how the response categories are split at each level of the nesting.

**Usage**

```
as.tree(x, ...)

## S3 method for class 'dichotomies'
as.tree(x, response = NULL, lobstr = FALSE, ...)

## S3 method for class 'continuationDichotomies'
as.tree(x, response = NULL, lobstr = FALSE, ...)
```

**Arguments**

x	A "dichotomies" or "continuationDichotomies" object.
...	additional arguments (currently unused).
response	Optional character string giving the name of the response variable, used as the root label of the tree. If NULL (default), the root is labeled "(response)".
lobstr	Logical. If FALSE (default), renders a 2-D ASCII tree with / and \ branch connectors, with each node centered above its two children. If TRUE, builds a nested list representation of the tree and renders it via <a href="#">tree</a> , which must be installed.

**Details**

The flat list of dichotomies is reconstructed into a binary tree by matching each dichotomy's domain (the union of its two sides) to the multi-level groups produced by earlier splits. Branch labels are taken from the named arguments to [dichotomy](#) when present, and are otherwise generated automatically as {level1, level2, ...}.

**Value**

Invisibly returns x; called for its side effect of printing.

**See Also**

[logits](#), [continuationLogits](#), [print.dichotomies](#) Other conversions: [as.matrix.dichotomies](#), [as.character.dichotomies](#)

**Examples**

```
## Womenlf: named group on one branch
comparisons <- logits(work = dichotomy("not.work",
                                     working = c("parttime", "fulltime")),
                    full = dichotomy("parttime", "fulltime"))
as.tree(comparisons, response = "partic")

## GSS: continuation logits for ordered education levels
cont <- continuationLogits(c("<highschool", "highschool",
                            "college", "graduate"))
as.tree(cont, response = "degree")

## gators data: Food choice
# create dichotomies
gators.dichots <- logits(d1=dichotomy("Other", c("Fish", "Invertebrates")),
                       d2=dichotomy("Fish", "Invertebrates"))
as.tree(gators.dichots, response = "Food")
as.tree(gators.dichots, response = "Food", lobstr = TRUE)
```

**Description**

These functions give compact summaries of a "nestedLogit" object

`glance` Construct a single row summaries for the dichotomies "nestedLogit" model.

`tidy` Summarizes the terms in "nestedLogit" model.

**Usage**

```
## S3 method for class 'nestedLogit'  
glance(x, ...)
```

```
## S3 method for class 'nestedLogit'  
tidy(x, ...)
```

**Arguments**

`x` an object of class "nestedLogit".  
`...` arguments to be passed down.

**Value**

- `glance` returns a tibble containing one row of fit statistics for each dichotomy, labeled response. See [glance](#) for details.
- `tidy` returns a tibble containing coefficient estimates and test statistics for the combinations of response and term. See [tidy](#) for details.

**See Also**

[nestedMethods](#), [glance](#), [tidy](#)

**Examples**

```
data("Women1f", package = "carData")  
m <- nestedLogit(partic ~ hincome + children,  
                 dichotomies = logits(work=dichotomy("not.work",  
                                                    working=c("parttime", "fulltime")),  
                                     full=dichotomy("parttime", "fulltime")),  
                 data=Women1f)  
  
# one-line summaries  
broom::glance(m)  
# coefficients and tests  
broom::tidy(m)
```

---

Effect.nestedLogit      *Effect Displays for Nested Logit Models*

---

### Description

Computes effects (in the sense of the **effects** package—see, in particular, [Effect](#))—for "nestedLogit" models, which then can be used with other functions in the **effects** package, for example, [predictorEffects](#) and to produce effect plots.

### Usage

```
## S3 method for class 'nestedLogit'  
Effect(  
  focal.predictors,  
  mod,  
  confidence.level = 0.95,  
  fixed.predictors = NULL,  
  ...  
)
```

### Arguments

`focal.predictors`  
a character vector of the names of one or more of the predictors in the model, for which the effect display should be computed.

`mod`  
a "nestedLogit" model object.

`confidence.level`  
for point-wise confidence bands around the effects (the default is 0.95).

`fixed.predictors`  
controls the values at which other predictors are fixed; see [Effect](#) for details; if NULL (the default), numeric predictors are set to their means, factors to their distribution in the data.

...  
optional arguments to be passed to the [Effect](#) method for binary logit models (fit by the [glm](#) function).

### Value

an object of class "effpoly" (see [Effect](#)).

### Author(s)

John Fox

## References

John Fox and Sanford Weisberg (2019). *An R Companion to Applied Regression*, 3rd Edition. Sage, Thousand Oaks, CA.

John Fox, Sanford Weisberg (2018). Visualizing Fit and Lack of Fit in Complex Regression Models with Predictor Effect Plots and Partial Residuals. *Journal of Statistical Software*, 87(9), 1-27.

## See Also

[Effect](#), [plot.effpoly](#), [predictorEffects](#)

## Examples

```
data("Womenlf", package = "carData")
comparisons <- logits(work=dichotomy("not.work",
                                     working=c("parttime", "fulltime")),
                    full=dichotomy("parttime", "fulltime"))
m <- nestedLogit(partic ~ hincome + children,
                 dichotomies = comparisons,
                 data=Womenlf)
peff.women <- effects::predictorEffects(m)
plot(peff.women)
plot(peff.women, axes=list(y=list(style="stacked")))
summary(peff.women)

dichots <- logits(AB_CD = dichotomy(c("A", "B"), c("C", "D")),
                 A_B   = dichotomy("A", "B"),
                 C_D   = dichotomy("C", "D"))
m.health <- nestedLogit(product4 ~ age + gender*household + position_level,
                       dichotomies = dichots, data = HealthInsurance)
eff.gen.hh <- effects::Effect(c("gender", "household"), m.health,
                             xlevels=list(household=0:7))
eff.gen.hh
plot(eff.gen.hh, axes=list(x=list(rug=FALSE)))
plot(eff.gen.hh, axes=list(x=list(rug=FALSE),
                             y=list(style="stacked")))
```

---

gators

*Alligator Food Choice*

---

## Description

Agresti (1996, p. 207) gives this data on 59 alligators sampled from a lake in Florida. It has the length of the alligator in meters and the primary food type found in the alligator's stomach. The food type was classified into three categories: "Fish", "Invertebrates", and "Other".

Of interest is whether or not the length of an alligator is associated with the primary food type. Does knowing the length of an alligator give us some indication about its primary food type? If so, how is length associated with the choice of food type?

**Usage**

```
data("gators", package = "nestedLogit")
```

**Format**

A data frame with 59 rows and 2 columns.

**food** Primary food type found in the alligator's stomach, a factor with levels "Other", "Fish", and "Invertebrates".

**length** Length of the alligator in meters, a numeric vector.

**Source**

Agresti, A. (1996). *An Introduction to Categorical Data Analysis*. Wiley.

**References**

An example using this from <https://data.library.virginia.edu/getting-started-with-multinomial-logit-models/>

**See Also**

[nestedLogit](#).

**Examples**

```
data(gators)
table(gators$food)
# average length of gators by food
with(gators, tapply(length, food, mean))

# create dichotomies
dichot <- logits(d1=dichotomy("Other", c("Fish", "Invertebrates")),
                d2=dichotomy("Fish", "Invertebrates"))

gators.nested <- nestedLogit(food ~ length,
                             dichotomies = dichot,
                             data = gators)

car::Anova(gators.nested)

# use the plot method
plot(gators.nested, x.var = "length")
```

**Description**

This data set is drawn from the U.S. General Social Survey (GSS) for years between 1972 and 2016.

**Usage**

```
data("GSS", package = "nestedLogit")
```

**Format**

A data frame with 44091 rows and 3 columns.

**parentdeg** A factor representing parents' attained level of education (highest "degree" obtained), recording the higher of mother's and father's education, with levels "<highschool", "highschool", "college", and "graduate".

**degree** The respondent's level of education, a factor with the same levels as parentdeg.

**year** The year of the survey, between 1972 and 2016.

**Source**

General Social Survey, NORC, The University of Chicago <https://www.norc.org/Research/Projects/Pages/general-social-survey.aspx>.

**See Also**

[nestedLogit](#).

**Examples**

```
round(100*with(GSS, prop.table(table(degree, parentdeg), 2)))
m.GSS <- nestedLogit(degree ~ parentdeg*year,
                    continuationLogits(c("<highschool", "highschool",
                                         "college", "graduate")),
                    data=GSS)
car::Anova(m.GSS)
summary(m.GSS)

# plot fitted probabilities
plot(m.GSS, x.var = "year",
      others = list(parentdeg = "<highschool"),
      lty = 1,
      label = TRUE)
plot(m.GSS, x.var = "year",
      others = list(parentdeg = "graduate"),
      lty = 1,
      label = TRUE)
```

---

 HealthInsurance

*Choice of Health Insurance Product*


---

### Description

A company recently introduced a new health insurance provider for its employees. At the beginning of the year the employees had to choose one of three (or four) different health plan products from this provider to best suit their needs.

This dataset was modified from its original source (McNulty, 2022) for the present purposes by adding a fourth choice, sampled randomly from the original three.

### Usage

```
data("HealthInsurance", package = "nestedLogit")
```

### Format

A data frame with 1448 rows and 7 columns.

**product** Choice among three products, a factor with levels "A", "B", and "C".

**product4** Choice among four products, a factor with levels "A", "B", "C", and "D".

**age** The age of the individual, in years.

**household** The number of people living with the individual in the same household.

**position\_level** Position level in the company at the time the choice was made, where 1 is the lowest level and 5 is the highest, a numeric vector.

**gender** The gender of the individual, a factor with levels "Female" and "Male".

**absent** The number of days the individual was absent from work in the year prior to the choice,

### Source

Originally taken from McNulty, K. (2022). *Handbook of Regression Modeling in People Analytics*, [https://peopleanalytics-regression-book.org/data/health\\_insurance.csv](https://peopleanalytics-regression-book.org/data/health_insurance.csv).

### See Also

[nestedLogit](#).

### Examples

```
lbinary <- logits(AB_CD = dichotomy(c("A", "B"), c("C", "D")),
                A_B = dichotomy("A", "B"),
                C_D = dichotomy("C", "D"))
as.matrix(lbinary)
health.nested <- nestedLogit(product4 ~ age + gender * household + position_level,
                             dichotomies = lbinary, data = HealthInsurance)
car::Anova(health.nested)
coef(health.nested)
```

---

models

*Extract Binary Logit Models from a nestedLogit Object*


---

### Description

Nested logit models represent an overall models for a polytomous response (>2 categories) by a set of binary logit models corresponding to nested dichotomies among the response categories. `models` is used to extract "glm" objects representing binary logit models from a "nestedLogit" object.

### Usage

```
models(model, select, as.list = FALSE)
```

```
## S3 method for class 'nestedLogit'
models(model, select, as.list = FALSE)
```

### Arguments

<code>model</code>	a "nestedLogit" model.
<code>select</code>	a numeric or character vector giving the number(s) or names(s) of one or more binary logit models to be extracted from <code>model</code> ; if absent, a list of all of the binary logits models in <code>model</code> is returned.
<code>as.list</code>	if TRUE (the default is FALSE) and one binary logit model is selected, return the "glm" object in a one-element named list; otherwise a single model is returned directly as a "glm" object; when more than one binary logit model is selected, the corresponding "glm" objects are <i>always</i> returned as a named list.

### Value

`model` returns either a single "glm" object (see [glm](#)) or a list of "glm" objects, each representing a binary logit model.

### Examples

```
data("Women1f", package = "carData")
comparisons <- logits(work=dichotomy("not.work",
                                     working=c("parttime", "fulltime")),
                    full=dichotomy("parttime", "fulltime"))
m <- nestedLogit(partic ~ hincome + children,
                dichotomies = comparisons,
                data=Women1f)

# extract both submodels, as a list
models(m, c("work", "full"))

# extract the binomial logit model for working vs. non-working
m_work <- models(m, "work")
```

```
# use that to plot residuals
plot(density(residuals(m_work)))

# or plot that model -- gives the 'regression quartet' for a glm()
op <- par(mfrow = c(2,2))
plot(m_work)
par(op)
```

---

nestedHypotheses      *Hypothesis-Testing and Related Methods for "nestedLogit" Objects*

---

## Description

Various methods for testing hypotheses about nested logit models.

**Anova** Calculates type-II or type-III analysis-of-variance tables for "nestedLogit" objects; see [Anova](#) in the **car** package.

**anova** Computes sequential analysis of variance (or deviance) tables for one or more fitted "nestedLogit" objects; see [anova](#).

**linearHypothesis** Computes Wald tests for linear hypotheses; see [linearHypothesis](#) in the **car** package.

**logLik** Returns the log-likelihood and degrees of freedom for the nested-dichotomies model. (and through it [AIC](#) and [BIC](#) model-comparison statistics).

## Usage

```
## S3 method for class 'nestedLogit'
Anova(mod, ...)

## S3 method for class 'Anova.nestedLogit'
print(x, ...)

## S3 method for class 'nestedLogit'
linearHypothesis(model, ...)

## S3 method for class 'nestedLogit'
anova(object, object2, ...)

## S3 method for class 'anova.nestedLogit'
print(x, ...)

## S3 method for class 'nestedLogit'
logLik(object, ...)
```

**Arguments**

... arguments to be passed down. In the case of `linearHypothesis`, the second argument is typically the `hypothesis.matrix`. See the Details section of [linearHypothesis](#). In the case of `anova`, additional sequential "nestedLogit" models.

`x`, `object`, `object2`, `mod`, `model`  
in most cases, an object of class "nestedLogit".

**Value**

- The `Anova` and `anova` methods return objects of class "Anova.nestedLogit" and "anova.nestedLogit", respectively, each of which contains a list of "anova" objects (see [anova](#)) and is usually printed.
- The `linearHypothesis` method is called for its side effect, printing the result of linear hypothesis tests, and invisibly returns NULL.
- The `logLik` method returns an object of class "logLik" (see [logLik](#)).

**Author(s)**

John Fox

**See Also**

[Anova](#), [anova](#), [linearHypothesis](#), [logLik](#), [AIC](#), [BIC](#)

**Examples**

```
# define continuation dichotomies for level of education
cont.dichots <- continuationLogits(c("<highschool",
                                   "highschool",
                                   "college",
                                   "graduate"))

# fit a nested model for the GSS data examining education degree in relation to parent & year
m <- nestedLogit(degree ~ parentdeg + year,
                 cont.dichots,
                 data=GSS)

# Anova and anova tests
car::Anova(m) # type-II (partial) tests

anova(update(m, . ~ . - year), m) # model comparison

# Wald test
car::linearHypothesis(m, c("parentdeghighschool", "parentdegcollege",
                          "parentdeggraduate"))

# log-likelihood, AIC, and BIC
logLik(m)
AIC(m)
BIC(m)
```

**Description**

Fit a related set of binary logit models via the `glm` function to nested dichotomies, comprising a model for the polytomy. A polytomous response with  $m$  categories can be analyzed using  $m - 1$  binary logit comparisons. When these comparisons are nested, the  $m - 1$  sub-models are statistically independent. Therefore, the likelihood chi-square statistics for the sub-models are additive and give overall tests for a model for the polytomy. This method was introduced by Fienberg (1980), and subsequently illustrated by Fox(2016) and Friendly & Meyer (2016).

`dichotomy` and `logits` are helper functions to construct the dichotomies.

`continuationLogits` constructs a set of  $m - 1$  logit comparisons, called continuation logits, for an ordered response. With  $m = 4$  levels, say, A, B, C, D, considered low to high: The first contrasts B, C, D against A. The second ignores A and contrasts C, D against B. The second ignores A, B and contrasts D against C.

**Usage**

```
nestedLogit(formula, dichotomies, data, subset = NULL, contrasts = NULL, ...)
logits(...)
dichotomy(...)
continuationLogits(levels, names, prefix = "above_")
```

**Arguments**

<code>formula</code>	a model formula with the polytomous response on the left-hand side and the usual linear-model-like specification on the right-hand side.
<code>dichotomies</code>	specification of the logits for the nested dichotomies, constructed by the <code>logits</code> and <code>dichotomy</code> functions, or <code>continuationLogits</code> . Alternatively, the dichotomies can be specified as a nested (i.e., recursive) list, the elements of which can be given optional names. See Details.
<code>data</code>	a data frame with the data for the model; unlike in most statistical modeling functions, the data argument is required. Cases with NAs in any of the variables appearing in the model formula will be removed with a Note message.
<code>subset</code>	a character string specifying an expression to fit the model to a subset of the data; the default, <code>NULL</code> , uses the full data set.
<code>contrasts</code>	an optional list of contrast specification for specific factors in the model; see <code>lm</code> for details.
<code>...</code>	for <code>nestedLogit</code> , optional named arguments to be passed to <code>glm</code> ; for <code>logits</code> , definitions of the nested logits—with each named argument specifying a dichotomy; for <code>dichotomy</code> , two character vectors giving the levels defining the dichotomy; the vectors can optionally be named.

levels	A character vector of set of levels of the variables or a number specifying the numbers of levels (in which case, uppercase letters will be use for the levels).
names	Names to be assigned to the dichotomies; if absent, names will be generated from the levels.
prefix	a character string (default: "above_") used as a prefix to the names of the continuation dichotomies.

## Details

A *dichotomy* for a categorical variable is a comparison of one subset of levels against another subset. A set of dichotomies is *nested*, if after an initial dichotomy, all subsequent ones are *within* the groups of levels lumped together in earlier ones. Nested dichotomies correspond to a binary tree of the successive divisions.

For example, for a 3-level response, a first dichotomy could be {A}, {B, C} and then the second one would be just {B}, {C}. Note that in the second dichotomy, observations with response A are treated as NA.

The function `dichotomy` constructs a *single* dichotomy in the required form, which is a list of length 2 containing two character vectors giving the levels defining the dichotomy. The function `logits` is used to create the set of dichotomies for a response factor. Alternatively, the nested dichotomies can be specified more compactly as a nested (i.e., recursive) list with optionally named elements; for example, `list(air="plane", ground=list(public=list("train", "bus"), private="car"))`.

The function `continuationLogits` provides a convenient way to generate all dichotomies for an ordered response. For an ordered response with  $m = 4$  levels, say, A, B, C, D, considered low to high: The dichotomy first contrasts B, C, D against A. The second ignores A and contrasts C, D against B. The second ignores A, B and contrasts D against C.

## Value

`nestedLogit` returns an object of class "nestedLogit" containing the following elements:

- `models`, a named list of (normally)  $m - 1$  "glm" objects, each a binary logit model for one of the  $m - 1$  nested dichotomies representing the  $m$ -level response.
- `formula`, the model formula for the nested logit models.
- `dichotomies`, the "dichotomies" object defining the nested dichotomies for the model.
- `data.name`, the name of the data set to which the model is fit, of class "name".
- `data`, the data set to which the model is fit.
- `subset`, a character representation of the subset argument or "NULL" if the argument isn't specified.
- `contrasts`, the contrasts argument or NULL if the argument isn't specified.
- `contrasts.print` a character representation of the contrasts argument or "NULL" if the argument isn't specified.

`logits` and `continuationLogits` return objects of class "dichotomies" and `c("continuationDichotomies" "dichotomies")`, respectively, which are two-elements lists, each element containing a list of two character vectors representing a dichotomy. `dichotomy` returns a list of two character vectors representing a dichotomy.

**Author(s)**

John Fox

**References**

S. Fienberg (1980). *The Analysis of Cross-Classified Categorical Data*, 2nd Edition, MIT Press, Section 6.6.

J. Fox (2016), *Applied Linear Regression and Generalized Linear Models*, 3rd Edition, Sage, Section 14.2.2.

J. Fox and S. Weisberg (2011), *An R Companion to Applied Regression*, 2nd Edition, Sage, Section 5.8.

M. Friendly and D. Meyers (2016), *Discrete Data Analysis with R*, CRC Press, Section 8.2.

**See Also**

[nestedMethods](#)

**Examples**

```
data("Women1f", package = "carData")

#' Use `logits()` and `dichotomy()` to specify the comparisons of interest
comparisons <- logits(work=dichotomy("not.work",
                                     working=c("parttime", "fulltime")),
                    full=dichotomy("parttime", "fulltime"))
print(comparisons)

m <- nestedLogit(partic ~ hincome + children,
                dichotomies = comparisons,
                data=Women1f)
print(summary(m))
print(car::Anova(m))
coef(m)

# equivalent;
nestedLogit(partic ~ hincome + children,
            dichotomies = list("not.work",
                              working=list("parttime", "fulltime")),
            data=Women1f)

# get predicted values
new <- expand.grid(hincome=seq(0, 45, length=10),
                 children=c("absent", "present"))
pred.nested <- predict(m, new)

# plot
op <- par(mfcol=c(1, 2), mar=c(4, 4, 3, 1) + 0.1)
plot(m, "hincome", list(children="absent"),
     xlab="Husband's Income", legend=FALSE)
plot(m, "hincome", list(children="present"),
```

```

      xlab="Husband's Income")
par(op)

continuationLogits(c("none", "gradeschool", "highschool", "college"))
continuationLogits(4)

```

---

 nestedMethods

---

*Methods for "nestedLogit" and Related Objects*


---

### Description

Various methods for processing "nestedLogit" and related objects. Most of these are the standard methods for a model-fitting function.

`coef`, `vcov` Return the coefficients and their variance-covariance matrix respectively.

`update` Re-fit a "nestedLogit" model with a change in any of the formula, dichotomies, data, subset, or contrasts, arguments.

`summary` Summarize a "nestedLogit" model, giving the summary for each binary logit model in the nested dichotomies.

`print` Print the model or a summary of the model.

`as.matrix`, `as.character`, `as.dichotomies` Coerce dichotomy-related objects to matrices, character vectors, and dichotomies objects.

### Usage

```

## S3 method for class 'nestedLogit'
print(x, ...)

## S3 method for class 'nestedLogit'
summary(object, ...)

## S3 method for class 'summary.nestedLogit'
print(x, ...)

## S3 method for class 'dichotomies'
print(x, ...)

## S3 method for class 'nestedLogit'
coef(object, as.matrix = TRUE, ...)

## S3 method for class 'nestedLogit'
vcov(object, as.matrix = FALSE, ...)

## S3 method for class 'nestedLogit'

```

```

update(object, formula, dichotomies, data, subset, contrasts, ...)

## S3 method for class 'dichotomies'
as.matrix(x, ...)

## S3 method for class 'dichotomies'
as.character(x, ...)

## S3 method for class 'continuationDichotomies'
as.matrix(x, ...)

as.dichotomies(x, ...)

## S3 method for class 'matrix'
as.dichotomies(x, ...)

```

### Arguments

x, object	in most cases, an object of class "nestedLogit".
...	arguments to be passed down.
as.matrix	if TRUE (the default for coef) return coefficients as a matrix with one column for each nested dichotomy, or coefficient covariances as a matrix with one row and column for each combination of dichotomies and coefficients; if FALSE (the default for vcov), return a list of coefficients or coefficient covariances with one element for each dichotomy.
formula	optional updated model formula.
dichotomies	optional updated dichotomies object.
data	optional updated data argument
subset	optional updated subset argument.
contrasts	optional updated contrasts argument.

### Value

- The coef and vcov methods return either matrices or lists of regression coefficients and their covariances, respectively.
- The update method returns an object of class "nestedLogit" (see [nestedLogit](#)) derived from the original nested-logit model.
- The summary method returns an object of class "summary.nestedLogit", which is a list of summaries of the [glm](#) objects that comprise the nested-dichotomies model; the object is normally printed.
- The methods for as.matrix, as.character, and as.dichotomies coerce various objects to matrices, character vectors, and dichotomies objects.
- The various print methods invisibly return their x arguments.

### Author(s)

John Fox and Michael Friendly

**See Also**

[nestedLogit](#), [predict.nestedLogit](#), [plot.nestedLogit](#), [glance.nestedLogit](#), [tidy.nestedLogit](#)

**Examples**

```
# define continuation dichotomies for level of education
cont.dichots <- continuationLogits(c("<highschool",
                                   "highschool",
                                   "college",
                                   "graduate"))

# Show dichotomies in various forms
print(cont.dichots)
as.matrix(cont.dichots)
as.character(cont.dichots)

# fit a nested model for the GSS data examining education degree in relation to parent & year
m <- nestedLogit(degree ~ parentdeg + year,
                 cont.dichots,
                 data=GSS)

coef(m) # coefficient estimates
sqrt(diag(vcov(m, as.matrix=TRUE))) # standard errors
print(m)
summary(m)
```

---

plot.nestedLogit

*Plotting Nested Logit Models*

---

**Description**

A `plot` method for "nestedLogit" objects produced by the `nestedLogit` function. Fitted probabilities under the model, or the corresponding logits are plotted for each level of the polytomous response variable, with one of the explanatory variables on the horizontal axis and other explanatory variables fixed to particular values. By default, a 95% pointwise confidence envelope is added to the plot.

**Usage**

```
## S3 method for class 'nestedLogit'
plot(
  x,
  x.var,
  others,
  n.x.values = 100L,
  scale = c("prob", "logit"),
  xlab = x.var,
```

```

ylab = NULL,
main,
cex.main = 1,
digits.main = getOption("digits") - 2L,
font.main = 1L,
pch = 1L:length(response.levels),
lwd = 3,
lty = 1L:length(response.levels),
col = (scales::hue_pal())(length(response.levels)),
legend = TRUE,
legend.inset = 0.01,
legend.location = "topleft",
legend.bty = "n",
conf.level = 0.95,
conf.alpha = 0.25,
label = FALSE,
label.x = "max",
label.cex = 1.25,
label.col = col,
...
)

```

### Arguments

<code>x</code>	an object of "nestedLogit" produced by <code>nestedLogit</code> .
<code>x.var</code>	quoted name of the variable to appear on the x-axis; if omitted, the first predictor in the model is used.
<code>others</code>	a named list of values for the other variables in the model, that is, other than <code>x.var</code> ; if any other predictor is omitted, it is set to an arbitrary value—the mean for a numeric predictor or the first level or value of a factor, character, or logical predictor; only one value may be specified for each variable in <code>others</code> .
<code>n.x.values</code>	the number of evenly spaced values of <code>x.var</code> at which to evaluate fitted probabilities to be plotted (default 100).
<code>scale</code>	character string; "prob" (the default) plots fitted probabilities on the y-axis; "logit" plots fitted log odds (logits), i.e., $\log(p/(1-p))$ .
<code>xlab</code>	label for the x-axis (defaults to the value of <code>x.var</code> ).
<code>ylab</code>	label for the y-axis (defaults to "Fitted Probability" when <code>scale = "prob"</code> and "Fitted Log Odds" when <code>scale = "logit"</code> ).
<code>main</code>	main title for the graph (if missing, constructed from the variables and values in <code>others</code> ).
<code>cex.main</code>	size of main title (see <code>par</code> ).
<code>digits.main</code>	number of digits to retain when rounding values for the main title.
<code>font.main</code>	font for main title (see <code>par</code> ).
<code>pch</code>	plotting characters (see <code>par</code> ).
<code>lwd</code>	line width (see <code>par</code> ).

lty	line types (see <a href="#">par</a> ).
col	line colors for the response levels (see <a href="#">par</a> ).
legend	if TRUE (the default), add a legend for the response levels to the graph. Ignored when label = TRUE.
legend.inset	default 0.01 (see <a href="#">legend</a> ).
legend.location	position of the legend (default "topleft", see <a href="#">legend</a> ).
legend.bty	the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n".
conf.level	the level for pointwise confidence envelopes around the predicted values; the default is 0.95. If NULL, the confidence envelopes are suppressed.
conf.alpha	the opacity of the confidence envelopes; the default is 0.3.
label	if TRUE, label the curves directly instead of using a legend. Default is FALSE.
label.x	where to place the label for each curve. Either a single string, "min" or "max" (applied to all curves), or a character vector of length equal to the number of response categories with each element being "min" or "max", e.g. label.x = c("min", "max", "max"). "min" places the label at the left end of the curve; "max" (the default) places it at the right end.
label.cex	character expansion factor for direct labels; default 1.25.
label.col	colors for direct labels; defaults to col, so labels match their curves. Supply a vector of length equal to the number of response categories to use different colors for the labels.
...	arguments to be passed to <a href="#">matplot</a> .

**Value**

NULL Used for its side-effect of producing a plot

**Author(s)**

John Fox, Michael Friendly

**See Also**

[nestedLogit](#), [matplot](#)

**Examples**

```
data("Womenlf", package = "carData")
m <- nestedLogit(partic ~ hincome + children,
                 logits(work=dichotomy("not.work", c("parttime", "fulltime")),
                       full=dichotomy("parttime", "fulltime")),
                 data=Womenlf)
plot(m, legend.location="top")

op <- par(mfcol=c(1, 2), mar=c(4, 4, 3, 1) + 0.1)
plot(m, "hincome", list(children="absent"),
```

```

      xlab="Husband's Income", legend=FALSE)
plot(m, "hincome", list(children="present"),
      xlab="Husband's Income")
par(op)

# Plot on the logit (log-odds) scale
plot(m, "hincome", list(children="absent"), scale = "logit",
      xlab = "Husband's Income")

# Gators example: direct curve labels instead of a legend
data("gators", package = "nestedLogit")
gators.nested <- nestedLogit(food ~ length,
  logits(d1 = dichotomy("Other", c("Fish", "Invertebrates")),
        d2 = dichotomy("Fish", "Invertebrates")),
  data = gators)

# All labels at the right end (default)
plot(gators.nested, x.var = "length", label = TRUE,
      xlab = "Alligator length (m)")

# Mixed placement: Other and Invertebrates labeled at left, Fish at right
# (food levels are: "Other", "Fish", "Invertebrates")
plot(gators.nested, x.var = "length", label = TRUE,
      label.x = c("min", "max", "min"),
      xlab = "Alligator length (m)")

```

---

predict.nestedLogit    *Predicted Probabilities and Logits for "nestedLogit" Models*

---

### Description

The predict and fitted methods compute predicted values from a fitted "nestedLogit" model. The confint method computes point-wise confidence limits for predicted response-category probabilities or logits.

predict, fitted Compute predicted response-category probabilities (or predicted logits for each binary logit model in the dichotomies) from a fitted "nestedLogit" model.

confint Compute point-wise confidence limits for predicted response-category probabilities or logits.

print **methods** Print predicted probabilities, logits, and their standard errors, with control over how many rows to display.

### Usage

```

## S3 method for class 'nestedLogit'
predict(object, newdata, model = c("nested", "dichotomies"), ...)

## S3 method for class 'predictNestedLogit'

```

```

print(x, n = min(10L, nrow(x$p)), ...)

## S3 method for class 'predictNestedLogit'
confint(
  object,
  parm = c("prob", "logit"),
  level = 0.95,
  conf.limits.logit = TRUE,
  ...
)

## S3 method for class 'predictDichotomies'
print(x, n = 10L, ...)

## S3 method for class 'nestedLogit'
fitted(object, model = c("nested", "dichotomies"), ...)

```

### Arguments

object	a fitted object of class "nestedLogit"; for confint, an object of class "predictNestedLogit".
newdata	a data frame containing combinations of values of the predictors at which fitted probabilities (or other quantities) are to be computed. If missing, the original data are used.
model	either "nested" (the default), in which case fitted probabilities under the nested logit model are returned, or "dichotomies", in which case <code>predict.glm</code> is invoked for each binary logit model fit to the nested dichotomies and a named list of the results is returned.
...	arguments to be passed down.
x	an object of class "predictNestedLogit" or "predictDichotomies".
n	an integer or "all" to control how many rows are printed for each of the predicted values.
parm	one of "prob" or "logit", indicating whether to generate confidence intervals for probabilities or logits of the responses.
level	confidence level, a number between 0 and 1; default is 0.95.
conf.limits.logit	logical; when parm = "prob", if TRUE (the default), confidence limits are computed on the logit scale and back-transformed to probabilities; if FALSE, Wald-type limits are computed directly on the probability scale.

### Details

The predict method provides predicted values for two representations of the model. `model = "nested"` (the default) gives the fitted probabilities for each of the response categories, along with the corresponding logits and standard errors of each. `model = "dichotomies"` gives the fitted log odds for each of the binary logit models in the nested dichotomies.

The fitted method (with no newdata) is equivalent to predict applied to the original data used to fit the model.

For the `confint` method with `parm = "prob"`, setting `conf.limits.logit = TRUE` (the default) computes confidence limits on the logit scale and back-transforms them to probabilities, which ensures that the limits lie in  $[0, 1]$ . Setting `conf.limits.logit = FALSE` computes Wald-type confidence intervals directly on the probability scale, which may extend outside  $[0, 1]$ .

### Value

- The `predict` and `fitted` methods return an object of class `"predictNestedLogit"` (when `model = "nested"`) or `"predictDichotomies"` (when `model = "dichotomies"`).  
A `"predictNestedLogit"` object is a list containing:
  - `p` a data frame of predicted probabilities, with one column per response category.
  - `logit` a data frame of predicted logits.
  - `se.p` a data frame of standard errors of predicted probabilities.
  - `se.logit` a data frame of standard errors of predicted logits.
  - `.data` the newdata data frame, if supplied.
 A `"predictDichotomies"` object is a named list of data frames, one per dichotomy, each produced by [predict.glm](#).
- The `confint` method returns a data frame of point estimates and confidence limits.
- The various `print` methods invisibly return their `x` arguments.

### Author(s)

John Fox and Michael Friendly

### See Also

[nestedLogit](#), [nestedMethods](#), [plot.nestedLogit](#), [as.data.frame.predictNestedLogit](#)

### Examples

```
# define continuation dichotomies for level of education
cont.dichots <- continuationLogits(c("<highschool",
                                   "highschool",
                                   "college",
                                   "graduate"))

# fit a nested model for the GSS data
m <- nestedLogit(degree ~ parentdeg + year,
                 cont.dichots,
                 data=GSS)

# predicted probabilities for first few cases
predict(m)

# predicted probabilities at specific values of predictors
new <- expand.grid(parentdeg=c("<highschool", "highschool",
                              "college", "graduate"),
                  year=c(1972, 2016))
```

```
fit <- predict(m, newdata=new)
cbind(new, fit)

# use fitted() -- equivalent to predict() on the original data
f <- fitted(m)

# predicted logits for each dichotomy
pred.dichot <- predict(m, newdata=new, model="dichotomies")
pred.dichot

# confidence intervals for predicted probabilities
fit.ci <- confint(fit)
head(fit.ci)

# confidence intervals for predicted logits
fit.ci.logit <- confint(fit, parm="logit")
head(fit.ci.logit)
```

# Index

- \* **datasets**
  - GSS, [9](#)
  - HealthInsurance, [10](#)
- \* **regression**
  - Effect.nestedLogit, [6](#)
  - nestedHypotheses, [12](#)
  - nestedLogit, [14](#)
  - nestedMethods, [17](#)
  - predict.nestedLogit, [22](#)
- AIC, [12](#), [13](#)
- Anova, [12](#), [13](#)
- anova, [12](#), [13](#)
- Anova.nestedLogit (nestedHypotheses), [12](#)
- anova.nestedLogit (nestedHypotheses), [12](#)
- as.character.dichotomies, [4](#)
- as.character.dichotomies (nestedMethods), [17](#)
- as.data.frame.predictNestedLogit, [2](#), [24](#)
- as.dichotomies (nestedMethods), [17](#)
- as.matrix.continuationDichotomies (nestedMethods), [17](#)
- as.matrix.dichotomies, [4](#)
- as.matrix.dichotomies (nestedMethods), [17](#)
- as.tree, [3](#)
- BIC, [12](#), [13](#)
- broomMethods, [5](#)
- coef.nestedLogit (nestedMethods), [17](#)
- confint.predictNestedLogit (predict.nestedLogit), [22](#)
- continuationLogits, [4](#)
- continuationLogits (nestedLogit), [14](#)
- dichotomy, [4](#)
- dichotomy (nestedLogit), [14](#)
- Effect, [6](#), [7](#)
- Effect.nestedLogit, [6](#)
- fitted.nestedLogit (predict.nestedLogit), [22](#)
- gators, [7](#)
- glance, [5](#)
- glance.nestedLogit, [19](#)
- glance.nestedLogit (broomMethods), [5](#)
- glm, [6](#), [11](#), [14](#), [18](#)
- GSS, [9](#)
- HealthInsurance, [10](#)
- legend, [21](#)
- linearHypothesis, [12](#), [13](#)
- linearHypothesis.nestedLogit (nestedHypotheses), [12](#)
- lm, [14](#)
- logits, [4](#)
- logits (nestedLogit), [14](#)
- logLik, [13](#)
- logLik.nestedLogit (nestedHypotheses), [12](#)
- make.names, [2](#)
- matplot, [21](#)
- models, [11](#)
- nestedHypotheses, [12](#)
- nestedLogit, [8–10](#), [14](#), [18–21](#), [24](#)
- nestedMethods, [5](#), [16](#), [17](#), [24](#)
- par, [20](#), [21](#)
- plot, [19](#)
- plot.effpoly, [7](#)
- plot.nestedLogit, [19](#), [19](#), [24](#)
- predict.glm, [23](#), [24](#)
- predict.nestedLogit, [2](#), [19](#), [22](#)
- predictorEffects, [6](#), [7](#)
- print.Anova.nestedLogit (nestedHypotheses), [12](#)

print.anova.nestedLogit  
    (nestedHypotheses), 12  
print.dichotomies, 4  
print.dichotomies (nestedMethods), 17  
print.nestedLogit (nestedMethods), 17  
print.predictDichotomies  
    (predict.nestedLogit), 22  
print.predictNestedLogit  
    (predict.nestedLogit), 22  
print.summary.nestedLogit  
    (nestedMethods), 17  
  
summary.nestedLogit (nestedMethods), 17  
  
tidy, 5  
tidy.nestedLogit, 19  
tidy.nestedLogit (broomMethods), 5  
tree, 4  
  
update.nestedLogit (nestedMethods), 17  
  
vcov.nestedLogit (nestedMethods), 17