Package 'hopit'

January 18, 2019

---

anova.hopit *LRT Tables*

Description

Compute likelihood rato test for two or more `hopit` objecs.

Usage

```
## S3 method for class 'hopit'
anova(object, ..., method = c("sequential", "with.first"),
direction = c("decreasing", "increasing"))
```

Arguments

object	an object containing the results returned by a `hopit`.
...	additional objects of the same type.
method	the method of model comparison. Choose `"sequential"` for 1-2, 2-3, 3-4, ... comparisons or `"with.first"` for 1-2, 1-3, 1-4, ... comparisons.
direction	determine if complexity of listed models is `"increasing"` or `"decreasing"` (default).

Value

`lrt.hopit` or `anova.hopit` objects.

Author(s)

Maciej J. Danko

See Also

print.anova.hopit, print.lrt.hopit, lrt.hopit, hopit.

1

---

boot_hopit_CI   *Calculating Confidence Intervals using percentile method*

Description

Calculating Confidence Intervals using percentile method

Usage

```
boot_hopit_CI(boot, alpha = 0.05, bounds = c("both", "lo", "up"))
```

Arguments

boot	boot object calculated by [boot_hopit](#) .
alpha	significance level.
bounds	one of "both", "lo", "up".

Author(s)

Maciej J. Danko

---

boot_hopit   *Bootstraping hopit model*

Description

Bootstraping hopit model

Usage

```
boot_hopit(model, data, func, nboot = 500, unlist = TRUE, ...)
```

Arguments

model	a fitted Hopit model.
data	data used to fit the model.
func	function to be bootstrapped of the form func(model, data, ...).
nboot	number of bootstrap replicates.
unlist	logical indicting if to unlist boot object.
...	other parameters passed to the func.

Author(s)

Maciej J. Danko

---

getCutPoints *Calcualte threshold cut-points using Jurges' method*

Description

Calcualte threshold cut-points using Jurges' method.

Usage

```
getCutPoints(model, subset = NULL, plotf = TRUE, mar = c(4, 4, 1, 1),
  oma = c(0, 0, 0, 0), XLab = "Health index", XLab.cex = 1.1,
  YLab = "Counts", YLab.cex = 1.1, decreasing.levels = TRUE,
  group.labels.type = c("middle", "border", "none"))
```

Arguments

model	a fitted hopit model.
subset	an optional vector specifying a subset of observations.
plotf	logical indicating if to plot the results.
mar, oma	see [par](#).
XLab, XLab.cex	label and size of the label for x axis.
YLab, YLab.cex	label and size of the label for y axis.
decreasing.levels	
	logical indicating if self-reported health classes are ordered in decreasing order.
group.labels.type	
	position of the legend. One of middel, border, or none.

Author(s)

Maciej J. Danko

---

getLevels *Calcualte adjusted health levels.*

Description

Calcualte adjusted health levels according to th Jurges' method.

Usage

```
getLevels(model, formula = model$thresh.formula,
  data = environment(model$thresh.formula), decreasing.levels = TRUE,
  sort.flag = FALSE, plotf = TRUE, sep = "_", mar = c(7, 2, 1.5, 0.5),
  oma = c(0, 3, 0, 0), YLab = "Fraction [%]", YLab.cex = 1.1,
  legbg = adjustcolor("white", alpha.f = 0.4), legbty = "o")
```

Arguments

`model`	a fitted `hopit` model.
`formula`	a formula containing the variables. It is by default set to threshold formula.
`data`	data used to fit the model.
`decreasing.levels`	logical indicating if self-reported health classes are ordered in increasing order.
`sort.flag`	logical indicating if to sort the levels.
`plotf`	logical indicating if to plot the results.
`sep`	separator for levels names.
`mar, oma`	see `par`.
`YLab, YLab.cex`	label and size of the label for y axis.
`legbg`	legend background color. See bg parameter in `legend`.
`legbty`	legend box type. See bty parameter in `legend`.

Author(s)

Maciej J. Danko

---

getTheta	*Extract Theta parameter from the* hopit *model*

Description

Extract Theta parameter from the `hopit` model

Usage

```
getTheta(model)
```

Arguments

`model`	fitted `hopit` model.

Author(s)

Maciej J. Danko

%c% *Check if one set is a subset of an another subset*

Description

Check if one set is a subset of an another subset

Usage

```
x %c% y
```

Arguments

x, y numeric vectors

Author(s)

Maciej J. Danko

%notc% *Not %c% function*

Description

Not %c% function

Usage

```
x %notc% y
```

Arguments

x, y numeric vectors

Author(s)

Maciej J. Danko

%notin%   *Not %in% function*

Description

Not %in% function

Usage

```
x %notin% y
```

Arguments

x, y   numeric vectors

Author(s)

Maciej J. Danko

---

healthsurvey   *Artificially generated health survey data*

Description

A dataset containing artificialy generated survey data

Usage

```
healthsurvey
```

Format

A data frame with 10000 rows and 11 variables:

ID personal indentification number

health reported health, 5 levels

diabetes has diabetes? Yes or No

obese has obese? Yes or No

IADL_problems problems in Instrumental Activities of Daily Living? Yes or No

hypertenssion has hypertenssion? Yes or No

high_cholesterol has high cholesterol? Yes or No

respiratory_problems has respirator problems? Yes or No

heart_atack_or_stroke had stroke or heart attack? Yes or No

poor_mobility has poor mobility? Yes or No

very_poor_grip Cannot perform grip strength? Yes or No

depression has depression? Yes or No

other_diseases has other diseases? Yes or No

sex sex/gender: woman or man

ageclass categorized age: [50,60), [60,70), [70,80), [80,120)

education two levels of education: primary or lower and secondary or higher

country country: X, Y, or Z

csw cross-sectional survey weights

psu primary statistical unit

ssu secondary statistical unit

Source

Data was randomly generated using probabilities of occurence of particular combiantion of diseases, coditions, sex, age, education, reported health, etc. The structure of the data and some probabilities were inspired by WAVE1 SHARE database (DOIs: 10.6103/SHARE.w1.600), see Börsch-Supan et al for methodological details (Börsch-Supan et al. 2013).

None of the records represent any true record/individual of SHARE database

References

Börsch-Supan A, Brandt M, Hunkler C, et al (2013) Data resource profile: The survey of health, ageing and retirement in europe (share). Int J Epidemiol 42:992–1001. doi: 10.1093/ije/dyt088

---

hopit.control   *Auxiliary for controlling* hopit *fitting*

Description

Auxiliary function for controlling hopit fitting. Use this function to set control parameters of the [hopit] and other related functions.

Usage

```
hopit.control(grad.eps = 3e-05, bgfs.maxit = 10000, cg.maxit = 10000,
  nlm.maxit = 150, bgfs.reltol = 5e-10, cg.reltol = 5e-10,
  nlm.gradtol = 1e-07, nlm.steptol = 1e-07, fit.methods = c("CG", "BFGS"),
  quick.fit = TRUE, trace = TRUE, thresh.start = -Inf,
  thresh.1.exp = FALSE, LL_out_val = -Inf)
```

Arguments

`grad.eps`	epsilon for numerical hessian function.
`bgfs.maxit, cg.maxit, nlm.maxit`	
	the maximum number of iterations. See [optim](#) and [nlm](#) for details.
`bgfs.reltol, cg.reltol`	
	relative convergence tolerance. See [optim](#) for details.
`nlm.gradtol, nlm.steptol`	
	tolerance at which the scaled gradient is considered close enough to zero and minimum allowable relative step length. See [nlm](#).
`fit.methods`	either 'CG' or 'BFGS'. See [optim](#).
`quick.fit`	logical, if TRUE extensive `nlm` optimization method is ignored and only BFGS and CG methods are run.
`trace`	logical, if to trace model fitting.
`LL_out_val, thresh.1.exp, thresh.start`	
	internal parameters under developement, do not change.

Author(s)

Maciej J. Danko

See Also

[hopit](#)

---

hopit	*Generelaized hierarchical ordered threshold models.*

Description

The ordered response data classifies a measure of interest into ordered categories collected during a survey. If the dependent variable is a happiness then a respondent typically answers a question: "Taking all things together, would you say you are ...? " and have some response options e.g. "very happy", "pretty happy", "not too happy", "very unhappy" \insertCiteLiao2005hopit. Similarly if interviewees are asked to evaluate their health in general (e.g. "Would you say your health is ...?") they may choose among several categories, e.g. very good, good, fair, bad, and very bad \insertCiteKing2004,Jurges2007,Rebelo2014hopit. In political sciences a respondent may be asked for an opinion about recent legislation (e.g. "Rate your feelings about the proposed legislation") and

asked to choose among several categories "strongly oppose", "mildly oppose", "indifferent", "mildly support", "strongly support" \insertCiteGreeneHensher2010hopit. It is easy to imagine other multi-level ordinal variables that might by use during the survey and to which methodology described below could be applied with.

Practically, it is assumed that when responding to a survey question about their general happiness, health, feeling, attitude or other status, participants assess their true value of this unobserved continuous variable, and project it to a provided discrete scale. The thresholds that each individual uses to categorize their true status into a specific response option may be affected by the choice of a reference group, earlier life experiences, and cross-cultural differences in using scales, and thus, may differ across individuals depending on their gender, age, cultural background, education, and personality traits, among other factors.

From the reporting-styles modeling perspective, one of the main tasks is to compute this continuous estimate of individuals' underlying, latent measure based on several specific characteristics of the considered response (e.g. health variables or happiness variables) and accounting also for variations in reporting across socio-demographic and cultural groups. More specifically, to build the latent, underlying measure a generalized hierarchical ordered threshold model is fitted, which regresses the reported status/attitude/feeling on two sets of independent variables \insertCite-Boes2006,Green2014hopit. When a dependent reported ordered variable is self-rated health status then the first set of variables (health variables) assesses individuals' specific aspects of health, and might include chronic conditions, mobility level, difficulties with a range of daily activities, performance on grip strength test, anthropometric measures, lifestyle behaviors, etc. Using the second set of independent variables (threshold variables), the model also adjusts for the differences across socio-demographic and cultural groups like cultural background, gender, age, education, etc. \insertCiteKing2004,Jurges2007hopit.

Ordered threshold models are used to fit ordered categorical dependent variables. The generalized ordered threshold models \insertCiteTerza1985,Boes2006,Green2014hopit are an extension to the ordered threshold models \insertCiteMcKelvey1975hopit. In the latter models the thresholds are constant, whereas generalized models allows thresholds to be dependent on covariates. \insertCite-GreeneHensher2010,Green2014;textualhopit pointed out that also thresholds must be ordered so that a model has a sense. This motivated Greene and coauthors to call this models *HOPIT*, which stands for hierarchical ordered probit models.

The fitted *hopit* model is used to analyse reporting styles. See [standardizeCoef](), [latentIndex](), [getCutPoints](), and [getLevels]().

Usage

```
hopit(latent.formula, thresh.formula = ~1, strata.formula = ~1, data,
  decreasing.levels, start = NULL, overdispersion = FALSE,
  design = list(), weights = NULL, link = c("probit", "logit"),
  control = list())
```

Arguments

latent.formula   formula used to model latent variable.

`thresh.formula`	formula used to model threshold variable. Any dependent variable (left side of "~") will be ignored.
`strata.formula`	formula used to model interactions between threshold and latent variables. Each term in this formula will be interacted with all latent variables then and added to the `latent.formula`. Each term in this formula will be also added to `threshold.formula`. Any dependent variable (left side of "~") will be ignored.
`data`	a data frame including all modeled variables.
`decreasing.levels`	
	logical indicating if self-reported health classes are ordered in decreasing order.
`start`	a vector with starting coefficient values in the form c(latent_parameters, threshold_lambdas, thre
`overdispersion`	logical indicting if to fit additional parameter theta modeling a variance of the error term.
`design`	an optional survey design. Use [svydesign](#) function to specify the design. The design cannot be speciffied together with parameter `weights`.
`weights`	an optional weights. Use design to construct survey weights.
`link`	the link function. The possible values are `"probit"` (default) and `"logit"`.
`control`	a list with control parameters. See [hopit.control](#).

Details

The function fits generelaized hierarchical ordered threshold models.

`latent.formula` models latent variable. if the response variable is self-rated health then latent measure can depend on different health conditions and diseases (latent variables are called health variables). Latent variables are modeled with parallel regression assumption. According to the assumption, coefficients, which describe the relationship between lowest and all higher response categories, are the same as those coefficients, which describe the relationship between another (e.g. adjacent) lowest and the remaining higher response categories. The predicted latent variable is modeled as a linear function of health variables and corresponding coefficients.

`thresh.formula` models threshold variable. The thresholds (cut points, `alpha`) are modeled by threshold variables gamma and intercepts `lambda`. It is assumed that they model contextual characteristics of the respondent (e.g. country, gender, age, etc.). Threshold variables are modeled without parallel regression assumption, thus each threshold is modeled by a variable independently \insertCiteBoes2006,Green2014hopit. hopit() function uses parametrization of tresholds proposed by \insertCiteJurges2007;textualhopit.

`decreasing.levels` it is the logical that determines the ordering of levels of the categorical response variable. It is always good to check first the ordering of the levels before starting (see example 1)

Value

a `hopit` object used by other functions and methods. The object is a list with following components:

control	a list with control parameters. See [hopit.control](#).
link	used link funtion.
hasdisp	logical, was overdispersion modeled?
latent.formula	used latent formula.
latent.mm	latent model matrix.
latent.terms	used latent variables.
thresh.formula	used threshold formula.
thresh.mm	threshold model matrix.
thresh.extd	threshold extended model matrix.
thresh.terms	used threshold variables.
thresh.no.cov	logical, are gamma parameters present?
parcount	3-element vector with number of parmeters for latent latent variable (beta), threshold intercept (lambda), and threshold covariates (gamma).
coef	vector with coefficients.
coef.ls	coefficients as a list.
start	vector with starting vlues of coefficients.
alpha	estimated individual-specific thresholds.
y_i	response variable.
y_latent_i	predicted latent measure.
Ey_i	predicted categorical response.
J	number of response levels.
N	number of observations.
deviance	deviance.
LL	log likelihood.
AIC	AIC for models without survey design.
vcov	variance-covariance matrix.
hessian	hessian matrix.
estfun	gradient of the log likelihood function at estimated coefficient values.
YYY1,YYY2,YY3	internal objects used for calcualtion of gradient and hessian functions.
use.weights,vcov.basic,glm.start,glm.start.ls	
	other internal objects.

Author(s)

Maciej J. Danko

References

\insertAllCited

See Also

coef.hopit, profile.hopit, hopit.control, anova.hopit, vcov.hopit, logLik.hopit, AIC.hopit, summary.hopit, svydesign,

For reporting styles analysis see:
standardizeCoef, latentIndex, getCutPoints, getLevels.

Examples

```
# DATA
data(healthsurvey)

# first determine the order of the dependent variable
levels(healthsurvey$health)

# Example 1 --------------------

# the order is decreasing (from the best health to the worst health)
# so we set: decreasing.levels = TRUE
# fitting the model:
model1 <- hopit(latent.formula = health ~ hypertenssion + high_cholesterol +
                  heart_atack_or_stroke + poor_mobility + very_poor_grip +
                  depression + respiratory_problems +
                  IADL_problems + obese + diabetes + other_diseases,
                thresh.formula = ~ sex + ageclass + country,
                decreasing.levels = TRUE,
                control = list(trace = FALSE),
                data = healthsurvey)

# summarize the fit:
summary(model1)

# extract parameters in a form of list
cm1 <- coef(model1, aslist = TRUE)

# names of returned coefficients
names(cm1)

# extracting latent health coefficients
cm1$latent.params

# Checking the fit
profile(model1)

# Example 2 --------------------

# incorporating survey design
design <- svydesign(ids = ~ country + psu, weights = healthsurvey$csw,
data = healthsurvey)

model2 <- hopit(latent.formula = health ~ hypertenssion + high_cholesterol +
                  heart_atack_or_stroke + poor_mobility +
```

```
                         very_poor_grip + depression + respiratory_problems +
                         IADL_problems + obese + diabetes + other_diseases,
                     thresh.formula = ~ sex + ageclass + country,
                     decreasing.levels = TRUE,
                     design = design,
                     control = list(trace = FALSE),
                     data = healthsurvey)

compare latent variables
cbind('No survey design' = coef(model1, aslist = TRUE)$latent.par,
'Has survey design' = coef(model2, aslist = TRUE)$latent.par)

# Example 3 --------------------

# using strata.formula
model3 <- hopit(latent.formula = health ~ hypertenssion + high_cholesterol +
                     heart_atack_or_stroke + poor_mobility + very_poor_grip +
                     depression + respiratory_problems +
                     IADL_problems + obese + diabetes + other_diseases,
                 thresh.formula = ~ sex + ageclass + country,
                 strata.formula = ~ sex,
                 decreasing.levels = TRUE,
                 control = list(trace = FALSE),
                 data = healthsurvey)

# print the model:
coef(model3, aslist = TRUE)$latent.params

# compare fit of model1 and model 3
# Likelihood Ratio Test
anova(model1, model3)

# AIC
AIC(model1, model3)

# Example 4 --------------------

# model with overdispersion
model4 <- hopit(latent.formula = health ~ hypertenssion + high_cholesterol +
                     heart_atack_or_stroke + poor_mobility + very_poor_grip +
                     depression + respiratory_problems +
                     IADL_problems + obese + diabetes + other_diseases,
                 thresh.formula = ~ sex + ageclass + country,
                 overdispersion = TRUE,
                 decreasing.levels = TRUE,
                 control = list(trace = FALSE),
                 data = healthsurvey)

# estimated variance of the error term:
getTheta(model4)

# compare fit of model1 and model4
# Likelihood Ratio Test
```

```
print(anova(model1, model4), short = TRUE)
```

latentIndex	*Calculate latent index*

Description

Calcualte latent index from the latent variable. It takes values from 0 to 1, where zero is prescribed to the worse predicted state (maximal observed value fo the latent variable) and 1 is prescribed to the best predicted health (minimal observed value fo the latent variable).

Usage

```
latentIndex(model, decreasing.levels = TRUE, subset = NULL, plotf = FALSE,
  response = c("data", "fitted", "Jurges"), ylab = "Latent index", ...)

healthIndex(model, decreasing.levels = TRUE, subset = NULL, plotf = FALSE,
  response = c("data", "fitted", "Jurges"), ylab = "Latent index", ...)
```

Arguments

model	a fitted hopit model.
decreasing.levels	
	logical indicating if self-reported (e.g. health) classes are ordered in decreasing order.
subset	an optional vector specifying a subset of observations.
plotf	logical indicating if to plot summary figure.
response	X axis plotting option, choose 'data' for raw responses and 'fitted' for model reclassified responses
ylab	a label of y axis.
...	futher parameters passedto the [plot](#) function.

Author(s)

Maciej J. Danko

standardizeCoef   *Standardization of coefficients*

Description

Calculate standardized coeficients - disability weights computed as the regression parameters from the generalised ordered probit model divided by the maximum possible range of its linear prediction. The range is calcualted as difference between maximum and minimum possible value of the latent variable given estimated parameters.

Usage

```
standardizeCoef(model, ordered = TRUE, plotf = FALSE, plotpval = FALSE,
  mar = c(15, 4, 1, 1), oma = c(0, 0, 0, 0), YLab = "Disability weight",
  YLab.cex = 1.1, namesf = identity, ...)

standardiseCoef(model, ordered = TRUE, plotf = FALSE, plotpval = FALSE,
  mar = c(15, 4, 1, 1), oma = c(0, 0, 0, 0), YLab = "Disability weight",
  YLab.cex = 1.1, namesf = identity, ...)

disabilityWeights(model, ordered = TRUE, plotf = FALSE, plotpval = FALSE,
  mar = c(15, 4, 1, 1), oma = c(0, 0, 0, 0), YLab = "Disability weight",
  YLab.cex = 1.1, namesf = identity, ...)
```

Arguments

model	a fitted `hopit` model.
ordered	logical indicating if to order the disability weights.
plotf	logical indicating if to plot results.
plotpval	logical indicating if to plot p-values.
mar, oma	see `par`.
YLab, YLab.cex	label and size of the label for y axis.
namesf	vectorof names of coeficients or one argument function that modifies names of coeficients.
...	arguments passed to `boxplot`.

Author(s)

Maciej J. Danko

See Also

`healthIndex`

Index