# Package 'hopit'

May 8, 2026

**Type** Package

**Title** Hierarchical Ordered Probit Models with Application to Reporting Heterogeneity

**Version** 0.11.6

**Depends** R (>= 3.5.0), survey (>= 4.1-1)

**Imports** MASS, Rcpp, graphics, stats, grDevices, questionr, parallel, Rdpack (>= 0.11.0)

**LinkingTo** Rcpp, RcppEigen

**Description** Self-reported health, happiness, attitudes, and other statuses or perceptions are often the subject of biases that may come from different sources. For example, the evaluation of an individual's own health may depend on previous medical diagnoses, functional status, and symptoms and signs of illness; as on well as life-style behaviors, including contextual social, gender, age-specific, linguistic and other cultural factors (Jylha 2009 <doi:10.1016/j.socscimed.2009.05.013>; Oksuzyan et al. 2019 <doi:10.1016/j.socscimed.2019.03.002>). The hopit package offers versatile functions for analyzing different self-reported ordinal variables, and for helping to estimate their biases. Specifically, the package provides the function to fit a generalized ordered probit model that regresses original self-reported status measures on two sets of independent variables (King et al. 2004 <doi:10.1017/S0003055403000881>; Jurges 2007 <doi:10.1002/hec.1134>; Oksuzyan et al. 2019 <doi:10.1016/j.socscimed.2019.03.002>). The first set of variables (e.g., health variables) included in the regression are individual statuses and characteristics that are directly related to the self-reported variable. In the case of self-reported health, these could be chronic conditions, mobility level, difficulties with daily activities, performance on grip strength tests, anthropometric measures, and lifestyle behaviors. The second set of independent variables (threshold variables) is used to model cut-points between adjacent self-reported response categories as functions of individual characteristics, such as gender, age group, education, and country (Oksuzyan et al. 2019 <doi:10.1016/j.socscimed.2019.03.002>). The model helps to adjust for specific socio-demographic and cultural differences in how the continuous latent health is projected onto the ordinal self-rated measure. The fitted model can be used to calculate an individual predicted latent status variable, a latent index, and standardized latent coefficients; and makes it possible to reclassify a categorical status measure that has been adjusted for inter-individual differences in reporting behavior.

**License** GPL-3

**Encoding** UTF-8

**LazyData** TRUE

**RoxygenNote** 7.2.2

**Suggests** testthat (>= 3.0.0), R.rsp (>= 0.43.0), usethis (>= 1.5.0), knitr (>= 1.20), rmarkdown (>= 1.12), qpdf, roxygen2 (>= 6.1.1)

**VignetteBuilder** R.rsp, knitr

**RdMacros** Rdpack

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Maciej J. Danko [aut, cre] (ORCID: <https://orcid.org/0000-0002-7924-9022>)

**Maintainer** Maciej J. Danko <Maciej.Danko@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-01-29 13:50:09 UTC

# Contents



---

`anova.hopit` *Likelihood Ratio Test Tables*

---

## Description

Perform the likelihood ratio test(s) for two or more `hopit` objects.

## Usage

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

## Arguments

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

## Value

a vector or a matrix with the results of the test(s).

## Author(s)

Maciej J. Danko

## See Also

`print.lrt.hopit`, `lrt.hopit`, `hopit`.

## Examples

```
# DATA
data(healthsurvey)

# the order of response levels decreases from the best health to
# the worst health; hence the hopit() parameter decreasing.levels
# is set to TRUE
levels(healthsurvey$health)

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

# fitting two nested models
model1 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                  heart_attack_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)

# a model with an interaction between hypertension and high_cholesterol
model2 <- hopit(latent.formula = health ~ hypertension * high_cholesterol +
                  heart_attack_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)

# a likelihood ratio test
lrt1 <- anova(model1, model2)
lrt1

# print results in a shorter form
print(lrt1, short = TRUE)

# or equivalently
lrt.hopit(model2, model1)

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

# fitting additional nested models
model3 <- hopit(latent.formula = health ~ hypertension * high_cholesterol +
                heart_attack_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)

model4 <- hopit(latent.formula = health ~ hypertension * high_cholesterol +
                heart_attack_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)

# sequential likelihood ratio tests
# model complexity increases so direction = "increasing"
anova(model1, model2, model3, model4,
      direction = "increasing", method = "sequential")

# likelihood ratio tests of the most complex model with the rest of the models
anova(model1, model2, model3, model4,
      direction = "increasing", method = "with.most.complex")

# likelihood ratio tests of the least complex model with the rest of the models
anova(model1, model2, model3, model4,
      direction = "increasing", method = "with.least.complex")
```

---

boot_hopit *Bootstrapping hopit model*

---

## Description

boot_hopit performs the bootstrap of a function dependent on a fitted model. In each of the bootstrap repetitions, a set of new model coefficients is drawn from the multivariate normal distribution, assuming the originally estimated model coefficients (see coef.hopit) as a mean and using the model variance-covariance matrix (see vcov.hopit). The drawn coefficients are then used to calculate the measure of interest using a function delivered by the func parameter.

## Usage

```
boot_hopit(
  model,
  func,
  data = model$frame,
  nboot = 500,
  unlist = TRUE,
  boot.only.latent = TRUE,
  parallel.flag = FALSE,
  parallel.nb_cores = NULL,
  parallel.packages = NULL,
  parallel.variables = NULL,
  robust.vcov,
  ...
)
```

## Arguments

| | |
|---|---|
| model | a fitted hopit model. |
| func | a function to be bootstrapped of the form func(model, ...). |
| data | data used to fit the model. |
| nboot | a number of bootstrap replicates. |
| unlist | a logical indicating whether to unlist the boot object. |
| boot.only.latent | a logical indicating whether to perform the bootstrap on latent variables only. |
| parallel.flag | a logical if to use parallel computations. |
| parallel.nb_cores | number of cores (<= number of CPU cores on the current host). |
| parallel.packages | list of packages needed to run "func". |
| parallel.variables | list of global variables and functions needed to run "func". |
| robust.vcov | see vcov.hopit. |
| ... | other parameters passed to the func. |

## Value

a list with bootstrapped elements.

## Author(s)

Maciej J. Danko

## See Also

percentile_CI, getLevels, getCutPoints, latentIndex, standardiseCoef, hopit.

## Examples

```
# DATA
data(healthsurvey)

# the order of response levels decreases from the best health to
# the worst health; hence the hopit() parameter decreasing.levels
# is set to TRUE
levels(healthsurvey$health)

# fit a model
model1 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                heart_attack_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)

# Example 1 ---------------------
# bootstrapping cut-points

# a function to be bootstrapped
cutpoints <-  function(model) getCutPoints(model)$cutpoints
B <- boot_hopit(model = model1, func = cutpoints, nboot = 100)

# calculate lower and upper bounds using the percentile method
cutpoints.CI <- percentile_CI(B)

# print estimated cutpoints and their confidence intervals
cutpoints(model1)
cutpoints.CI

# Example 2 ---------------------
# bootstrapping differences in health levels

# a function to be bootstrapped
diff_BadHealth <- function(model) {
  hl <- getLevels(model = model, formula=~ sex + ageclass, sep=' ')
  hl$original[,1] + hl$original[,2] - hl$adjusted[,1]- hl$adjusted[,2]
}

# estimate the difference
est.org <- diff_BadHealth(model = model1)
```

```
# perform the bootstrap
B <- boot_hopit(model = model1, func = diff_BadHealth, nboot = 100)

# calculate lower and upper bounds using the percentile method
est.CI <- percentile_CI(B)

# plot the difference and its (asymmetrical) confidence intervals
pmar <- par('mar'); par(mar = c(9.5,pmar[2:4]))
m <- max(abs(est.CI))
pos <- barplot(est.org, names.arg = names(est.org), las = 3,
               ylab = 'Original - Adjusted',
               ylim=c(-m, m), density = 20, angle = c(45, -45),
               col = c('blue', 'orange'))
for (k in seq_along(pos)) lines(c(pos[k,1],pos[k,1]),
                                est.CI[,k], lwd = 2, col = 2)
abline(h = 0); box(); par(mar = pmar)
```

---

`getCutPoints` *Calculate the threshold cut-points and individual adjusted responses using Jurges' method*

---

## Description

Calculate the threshold cut-points and individual adjusted responses using Jurges' method

## Usage

```
getCutPoints(model, decreasing.levels = model$decreasing.levels, subset = NULL)
```

## Arguments

| | |
|---|---|
| `model` | a fitted `hopit` model. |
| `decreasing.levels` | a logical indicating whether self-reported health classes are ordered in increasing order. |
| `subset` | an optional vector specifying a subset of observations. |

## Value

a list with the following components:

| | |
|---|---|
| `cutpoints` | cut-points for the adjusted categorical response levels with the corresponding percentiles of the latent index. |
| `adjusted.levels` | adjusted categorical response levels for each individual. |

## Author(s)

Maciej J. Danko

## References

Jurges H (2007). "True health vs response styles: exploring cross-country differences in self-reported health." *Health Economics*, **16**(2), 163-178. doi:10.1002/hec.1134.

Oksuzyan A, Danko MJ, Caputo J, Jasilionis D, Shkolnikov VM (2019). "Is the story about sensitive women and stoical men true? Gender differences in health after adjustment for reporting behavior." *Social Science & Medicine*, **228**, 41-50. doi:10.1016/j.socscimed.2019.03.002.

## See Also

latentIndex, standardiseCoef, getLevels, hopit.

## Examples

```
# DATA
data(healthsurvey)

# the order of response levels decreases from the best health to
# the worst health; hence the hopit() parameter decreasing.levels
# is set to TRUE
levels(healthsurvey$health)

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

# fit a model
model1 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                heart_attack_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)

# calculate the health index cut-points
z <- getCutPoints(model = model1)
z$cutpoints

plot(z)

# tabulate the adjusted health levels for individuals (Jurges method):
rev(table(z$adjusted.levels))

# tabulate the original health levels for individuals
table(model1$y_i)

# tabulate the predicted health levels
```

```
table(model1$Ey_i)
```

---

`getLevels` *Summarize the adjusted and the original self-rated response levels*

---

## Description

Summarize the adjusted and the original self-rated response levels.

## Usage

```
getLevels(
  model,
  formula = model$thresh.formula,
  data = model$frame,
  sep = "_",
  decreasing.levels = model$decreasing.levels,
  sort.flag = FALSE,
  weight.original = TRUE
)
```

## Arguments

| | |
|---|---|
| `model` | a fitted `hopit` model. |
| `formula` | a formula containing the grouping variables. It is by default set to threshold formula. |
| `data` | data used to fit the model. |
| `sep` | a separator for the level names. |
| `decreasing.levels` | a logical indicating whether self-reported health classes are ordered in increasing order. |
| `sort.flag` | a logical indicating whether to sort the levels. |
| `weight.original` | a logical indicating if use survey weights for calcualtion of original responses. |

## Value

a list with the following components:

| | |
|---|---|
| `original` | frequencies of original response levels for selected groups/categories. |
| `adjusted` | frequencies of adjusted response levels (Jurges 2007 method) for selected groups/categories. |
| `N.original` | the number of original response levels for selected groups/categories. |
| `N.adjusted` | the number of adjusted response levels (Jurges 2007 method) for selected groups/categories. |
| `categories` | selected groups/categories used in summary. |
| `tab` | an original vs. an adjusted contingency table. |

mat — a matrix with columns: grouping variables, original response levels, adjusted response levels. Each row corresponds to a single individual from the data used to fit the model.

## Author(s)

Maciej J. Danko

## References

Jurges H (2007). "True health vs response styles: exploring cross-country differences in self-reported health." *Health Economics*, **16**(2), 163-178. doi:10.1002/hec.1134.

Oksuzyan A, Danko MJ, Caputo J, Jasilionis D, Shkolnikov VM (2019). "Is the story about sensitive women and stoical men true? Gender differences in health after adjustment for reporting behavior." *Social Science & Medicine*, **228**, 41-50. doi:10.1016/j.socscimed.2019.03.002.

## See Also

getCutPoints, latentIndex, standardiseCoef, hopit.

## Examples

```
# DATA
data(healthsurvey)

# the order of response levels decreases from the best health to
# the worst health; hence the hopit() parameter decreasing.levels
# is set to TRUE
levels(healthsurvey$health)

# fit a model
model1 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                heart_attack_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)

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

# calculate a summary by country
hl <- getLevels(model=model1, formula=~ country, sep=' ')
plot(hl, las=1, mar = c(3,2,1.5,0.5))

# differences between frequencies of original and adjusted health levels
round(100*(hl$original - hl$adjusted),2)

# extract good and bad health levels (combined levels)
Org <- cbind(bad = rowSums(hl$original[,1:2]),
```

```
            good = rowSums(hl$original[,4:5]))
Adj <- cbind(bad = rowSums(hl$adjusted[,1:2]),
            good = rowSums(hl$adjusted[,4:5]))
round(100*(Org - Adj),2)

# plot the differences
barplot(t(Org - Adj), beside = TRUE, density = 20, angle = c(-45, 45),
        col = c('pink4', 'green2'),
        ylab = 'Original - adjusted reported health frequencies')
abline(h = 0); box()
legend('top', c('Bad health','Good health'),
       density = 20, angle = c(-45, 45),
       fill = c('pink4', 'green2'), bty = 'n', cex = 1.2)

# in country X, bad health seems to be over-reported while good health
# is under-reported; in country Z, good health is highly over-reported.

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

# summary by gender and age
hl <- getLevels(model = model1, formula=~ sex + ageclass, sep=' ')
plot(hl)

# differences between frequencies of original and adjusted health levels
round(100*(hl$original - hl$adjusted),2)

# extract good health levels (combined "Very good" and "Excellent" levels)
Org <- rowSums(hl$original[,4:5])
Adj <- rowSums(hl$adjusted[,4:5])
round(100*(Org - Adj),2)

pmar <- par('mar'); par(mar = c(9.5, pmar[2:4]))
barplot(Org-Adj,
        ylab = 'Original - adjusted reported good health frequencies',
        las = 3,
        density = 20, angle = c(45, -45), col = c('blue', 'orange'))
abline(h = 0); box(); par(mar = pmar)
legend('top', c('Man','Woman'), density = 20, angle = c(-45, 45),
       fill = c('blue', 'orange'), bty = 'n', cex = 1.2)

# results show that women in general tend to over-report good health,
# while men aged 50-59 greatly under-report good health.

# more examples can be found in the description of the boot_hopit() function.
```

---

`healthsurvey` *Artificially generated health survey data*

---

## Description

A dataset containing artificially generated survey data

## Usage

```
healthsurvey
```

## Format

A data frame with 10000 rows and 11 variables:

**ID** personal identification number.

**health** reported health, 5 levels.

**diabetes** has diabetes? "yes" or "no".

**obese** is obese? "yes" or "no".

**IADL_problems** has problems with Instrumental Activities of Daily Living? "yes" or "no".

**hypertension** has hypertension? "yes" or "no".

**high_cholesterol** has high cholesterol? "yes" or "no".

**respiratory_problems** has respiratory problems? "yes" or "no".

**heart_attack_or_stroke** had a stroke or a heart attack? "yes" or "no".

**poor_mobility** has poor mobility? "yes" or "no".

**very_poor_grip** cannot perform grip strength test? "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 ("prim-") and secondary or higher ("sec+".

**country** country: "X", "Y", or "Z".

**csw** cross-sectional survey weights.

**psu** primary statistical unit.

## Source

`healthsurvey` is a completely artificial data set simulated using distributions of some major health and socio-demographic characteristics. The distributions and the data structure are roughly based on the WAVE1 SHARE database (DOIs: 10.6103/SHARE.w1.600); see (Borsch-Supan et al. 2013) for technical details. None of the records represent any part of the true data.

The SHARE data collection has been primarily funded by the European Commission through FP5 (QLK6-CT-2001-00360), FP6 (SHARE-I3: RII-CT-2006-062193, COMPARE: CIT5-CT-2005-028857, SHARELIFE: CIT4-CT-2006-028812) and FP7 (SHARE-PREP: N°211909, SHARE-LEAP: N°227822, SHARE M4: N°261982). Additional funding from the German Ministry of Education and Research, the Max Planck Society for the Advancement of Science, the U.S. National Institute on Aging (U01_AG09740-13S2, P01_AG005842, P01_AG08291, P30_AG12815, R21_AG025169, Y1-AG-4553-01, IAG_BSR06-11, OGHA_04-064, HHSN271201300071C) and from various national funding sources is gratefully acknowledged (see www.share-project.org).

## References

Borsch-Supan A, Brandt M, Hunkler C, Kneip T, Korbmacher J, Malter F, Schaan B, Stuck S, Zuber S (2013). "Data Resource Profile: The Survey of Health, Ageing and Retirement in Europe (SHARE)." *International Journal of Epidemiology*, **42**(4), 992-1001. doi:10.1093/ije/dyt088.

## Examples

```
# load *healthsurvey* dataset
data(healthsurvey)

# horizontal view of the dataset (omitting ID)
print(t(healthsurvey[1:6,-1]), quote=FALSE, na.print='NA', right=TRUE)
```

---

`hopit` *Generalized hierarchical ordered threshold models.*

---

## Description

The ordered response data classify a measure of interest into ordered categories collected during a survey. For example, if the dependent variable is a happiness rating, a respondent typically answers a question such as: "Taking all things together, would you say you are ... ?" and then selects from response options along the lines of: "very happy", "pretty happy", "not too happy", and "very unhappy" (Liao et al. 2005). Similarly, if interviewees are asked to evaluate their health in general (e.g., "Would you say your health is ... ?") they, can typically choose among several categories, such as "very good", "good", "fair", "bad", and "very bad" (King et al. 2004; Jurges 2007; Rebelo and Pereira 2014; Oksuzyan et al. 2019). In political science, 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 categories like: "strongly oppose", "mildly oppose", "indifferent", "mildly support", and "strongly support" (Greene and Hensher 2010). It is easy to imagine other multi-level ordinal variables that might be used during a survey and to which the methodology described below could be applied.

In practice, it is assumed that when responding to a survey question about their general happiness, health, feelings, attitudes or other status, participants are assessing their true value of this unobserved continuous variable, and project it onto the discrete scale provided. The thresholds that individuals use to categorize their true status by selecting a specific response option may be affected by the reference group chosen, their earlier life experiences, and cross-cultural differences in using scales. Thus, the responses of individuals may differ depending on their gender, age, cultural background, education, and personality traits; among other factors (King et al. 2004; Jurges 2007; Oksuzyan et al. 2019).
From the perspective of reporting behavior modeling, one of the main tasks researchers face is to compute this continuous estimate of the underlying, latent measures of individuals based on several specific characteristics of the responses considered (e.g., health variables or happiness variables), and to account for variations in reporting across socio-demographic and cultural groups. More specifically, to build a latent, underlying measure, a generalized hierarchical ordered threshold model is fitted that regresses the reported status/attitude/feeling on two sets of independent variables (Boes and Winkelmann 2006; Greene et al. 2014). When the dependent reported ordered

variable is self-rated health status, then the first set of variables – i.e., health variables – assess specific aspects of individuals' health, such as measures of chronic conditions, mobility, difficulties with a range of daily activities, grip strength, anthropometric characteristics, and lifestyle behaviors. Using the second set of independent variables (threshold variables), the model also adjusts for differences across socio-demographic and cultural groups, such as differences in cultural background, gender, age, and education (King et al. 2004; Jurges 2007; Oksuzyan et al. 2019).

Ordered threshold models are used to fit ordered categorical dependent variables. The generalized ordered threshold models (Ierza 1985; Boes and Winkelmann 2006; Greene et al. 2014) are an extension of the ordered threshold models (McKelvey and Zavoina 1975). Whereas in the latter models, the thresholds are constant, in the generalized models the thresholds are allowed to be dependent on covariates. Greene and Hensher (2010); Greene et al. (2014) pointed out that for a model to make sense, the thresholds must also be ordered. This observation motivated Greene and coauthors to call these models *HOPIT*, which stands for hierarchical ordered probit models.

The fitted *hopit* model is used to analyze heterogeneity in reporting behavior. See `standardizeCoef`, `latentIndex`, `getCutPoints`, `getLevels`, and `boot_hopit`.

## Usage

```
hopit(
  latent.formula,
  thresh.formula = ~1,
  data,
  decreasing.levels,
  start = NULL,
  fit.sigma = FALSE,
  design = list(),
  weights = NULL,
  link = c("probit", "logit"),
  control = list(),
  na.action = na.fail
)
```

## Arguments

| | |
|---|---|
| `latent.formula` | a formula used to model the latent variable. It should not contain any threshold variable. To specify the interactions between the latent and the threshold variables, see details. |
| `thresh.formula` | a formula used to model the threshold variable. It should not contain any latent variable. To specify interactions between the latent and the threshold variables, see details. Any dependent variable (left side of "~" in the formula) will be ignored. |
| `data` | a data frame that includes all modeled variables. |
| `decreasing.levels` | a logical indicating whether self-reported health classes are ordered in decreasing order. |

| | |
|---|---|
| start | a vector with starting coefficient values in the form c(latent_parameters, threshold_lambdas, threshold_gammas) or c(latent_parameters, threshold_lambdas, threshold_gammas, logSigma) if the fit.sigma == TRUE. |
| fit.sigma | a logical indicating whether to fit an additional parameter sigma, which models a standard deviation of the error term (e.g., the standard deviation of the cumulative normal distribution in the probit model). |
| design | an optional survey design. Use the svydesign function to specify the design. The design cannot be specified together with parameter weights. |
| weights | optional model weights. Use design parameter to construct survey weights. |
| link | a link function. The possible values are "probit" (default) and "logit". |
| control | a list with control parameters. See hopit.control. |
| na.action | a function that indicates what should happen when the data contain NAs. The default is na.fail, which generates an error if any missing value is found. The alternative is na.omit (or na.exclude equivalently), which removes rows with missing values from the data. Using na.pass will lead to an error. |

## Details

The function fits generalized hierarchical ordered threshold models.

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

thresh.formula models the threshold variable. The thresholds (cut-points, alpha) are modeled by the threshold variables gamma and the intercepts lambda. It is assumed that they model the contextual characteristics of the respondent (e.g., country, gender, and age). The threshold variables are modeled without the parallel regression assumption; thus, each threshold is modeled by a variable independently (Boes and Winkelmann 2006; Greene et al. 2014). The hopit() function uses the parameterization of thresholds proposed by Jurges (2007).

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

It is possible to model the interactions, including interactions between the latent and the threshold variables. The interactions added to the latent formula only model the latent measure, and the interactions modeled in the threshold formula only model the thresholds. The general rule for modeling any kind of interaction is to use "*" to specify interactions within a latent (or threshold) formula and to use ':' to specify interactions between the latent and the threshold variables. In the latter case, the main effects of an interaction must also be specified; i.e., the main latent effects must be specified

in the latent formula, and the main threshold effect must be speciffied in the threshold formula. See also `Example 3` below.

For more details, please see the package vignette, which is also available under this link: vig_hopit.pdf

## Value

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

| | |
|---|---|
| `control` | a list with control parameters. See `hopit.control`. |
| `link` | a link function used. |
| `hasdisp` | a logical indicating whether fit.sigma was modeled. |
| `use.weights` | a logical indicating whether any weights were used. |
| `weights` | a vector with model weights. |
| `frame` | a model frame. |
| `latent.formula` | a latent formula used to fit the model. |
| `latent.mm` | a latent model matrix. |
| `latent.terms` | latent variables used, and their interactions. |
| `cross.inter.latent` | a part of the latent formula used for modeling cross-interactions in the latent model |
| `thresh.formula` | a threshold formula used to fit the model. |
| `thresh.mm` | a threshold model matrix. |
| `thresh.extd` | an extended threshold model matrix. |
| `thresh.terms` | threshold variables used, and their interactions. |
| `cross.inter.thresh` | a part of the threshold formula used for modeling cross-interactions in the threshold model |
| `thresh.no.cov` | a logical indicating whether gamma parameters are present. |
| `parcount` | a 3-element vector with a number of parameters for the latent variables (beta), the threshold intercepts (lambda), and the threshold covariates (gamma). |
| `coef` | a vector with model coefficients. |
| `coef.ls` | model coefficients as a list. |
| `start` | a vector with the starting values of the coefficients. |
| `alpha` | estimated individual-specific thresholds. |
| `y_i` | a vector with individual responses - the response variable. |
| `y_latent_i` | a vector with predicted latent measures for each individual. |
| `Ey_i` | a vector with predicted categorical responses for each individual. |
| `J` | a number of response levels. |
| `N` | a number of observations. |

| | |
|---|---|
| `deviance` | a deviance. |
| `LL` | a log likelihood. |
| `AIC` | an AIC for models without a survey design. |
| `vcov` | a variance-covariance matrix. |
| `vcov.basic` | a variance-covariance matrix that ignores the survey design. |
| `hessian` | a Hessian matrix. |
| `estfun` | a gradient (a vector of partial derivatives) of the log likelihood function at the estimated coefficient values. |
| `YYY1`, `YYY2`, `YYY3` | an internal objects used for the calculation of gradient and Hessian functions. |

## Author(s)

Maciej J. Danko

## References

Boes S, Winkelmann R (2006). "Ordered response models." *Allgemeines Statistisches Archiv*, **90**(1), 167–181. ISSN 1614-0176, doi:10.1007/s101820060228y.

Greene W, Harris MN, Hollingsworth B, Weterings TA (2014). "Heterogeneity in Ordered Choice Models: A Review with Applications to Self-Assessed Health." *Journal of Economic Surveys*, **28**(1), 109-133. doi:10.1111/joes.12002.

Greene W, Hensher D (2010). *Modeling Ordered Choices*. Cambridge University Press.

Ierza JV (1985). "Ordinal probit: A generalization." *Communications in Statistics - Theory and Methods*, **14**(1), 1-11. ISSN 0361-0926, doi:10.1080/03610928508828893.

Jurges H (2007). "True health vs response styles: exploring cross-country differences in self-reported health." *Health Economics*, **16**(2), 163-178. doi:10.1002/hec.1134.

King G, Murray CJL, Salomon JA, Tandon A (2004). "Enhancing the Validity and Cross-Cultural Comparability of Measurement in Survey Research." *American Political Science Review*, **98**(1), 191–207. doi:10.1017/S000305540400108X.

Liao P, Fu Y, Yi C (2005). "Perceived quality of life in Taiwan and Hong Kong: an intra-culture comparison." *Journal of Happiness Studies*, **6**(1), 43–67. ISSN 1573-7780, doi:10.1007/s10902-00417536.

McKelvey RD, Zavoina W (1975). "A Statistical Model for the Analysis of Ordinal Level Dependent Variables." *Journal of Mathematical Sociology*, **4**(1), 103–120.

Oksuzyan A, Danko MJ, Caputo J, Jasilionis D, Shkolnikov VM (2019). "Is the story about sensitive women and stoical men true? Gender differences in health after adjustment for reporting behavior." *Social Science & Medicine*, **228**, 41-50. doi:10.1016/j.socscimed.2019.03.002.

Rebelo LP, Pereira NS (2014). "Assessing health endowment, access and choice determinants:

Impact on retired Europeans' (in)activity and quality of life." *Social Indicators Research*, **119**(3), 1411-1446. doi:10.1007/s1120501305421.

## See Also

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

For heterogeneity in reporting behavior analysis see:
standardizeCoef, latentIndex, getCutPoints, getLevels, boot_hopit,

## Examples

```
# DATA
data(healthsurvey)

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

# the order of response levels decreases from the best health to
# the worst health; hence the hopit() parameter decreasing.levels
# is set to TRUE

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

# fitting the model:
model1 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                  heart_attack_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 the form of a list
cm1 <- coef(model1, aslist = TRUE)

# names of the returned coefficients
names(cm1)

# extract the latent health coefficients
cm1$latent.params

# check the fit

profile(model1)

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

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

model2 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                  heart_attack_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 the latent variables
cbind('No survey design' = coef(model1, aslist = TRUE)$latent.par,
'Has survey design' = coef(model2, aslist = TRUE)$latent.par)


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

# defining the interactions between the threshold and the latent variables

# correctly defined interactions:
model3 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                heart_attack_or_stroke + poor_mobility * very_poor_grip +
                depression + respiratory_problems +
                IADL_problems + obese + diabetes + other_diseases +
                sex : depression + sex : diabetes + ageclass:obese,
              thresh.formula = ~ sex * ageclass + country + sex : obese,
              decreasing.levels = TRUE,
              control = list(trace = FALSE),
              data = healthsurvey)

## Not run:
# badly defined interactions:

# 1) lack of a main effect of "other_diseases" in any formula
# it can be solved by adding " + other_diseases" to the latent formula
model3a <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                heart_attack_or_stroke + poor_mobility + very_poor_grip +
                depression + respiratory_problems +
                IADL_problems + obese + diabetes + other_diseases : sex,
              thresh.formula = ~ sex + ageclass + country,
              decreasing.levels = TRUE,
              control = list(trace = FALSE),
              data = healthsurvey)

# 2) the main effect of sex is present in both formulas.
# it can be solved by replacing "*" with ":" in "other_diseases * sex"
model3b <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
```

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


## End(Not run)
# Example 4 ---------------------


# construct a naive continuous variable:
hs <- healthsurvey
hs$cont_var <- sample(5000:5020,nrow(hs),replace=TRUE)

latent.formula = health ~ hypertension + high_cholesterol +
  heart_attack_or_stroke + poor_mobility + very_poor_grip +
  depression + respiratory_problems +
  IADL_problems + obese + diabetes + other_diseases

# in some cases, when continuous variables are used, the hopit:::get.hopit.start() function
# do not find starting parameters (R version 3.4.4 (2018-03-15)):
## Not run:
model4 <- hopit(latent.formula = latent.formula,
                thresh.formula = ~ sex + cont_var,
                decreasing.levels = TRUE,
                data = hs)

## End(Not run)
# one of the solutions is to transform one or more continuous variables:
hs$cont_var_t <- hs$cont_var-min(hs$cont_var)

model4b <- hopit(latent.formula = latent.formula,
                 thresh.formula = ~ sex + cont_var_t,
                 decreasing.levels = TRUE,
                 data = hs)

# this can also be done automatically using the the control parameter
model4c <- hopit(latent.formula = latent.formula,
                 thresh.formula = ~ sex + cont_var,
                 decreasing.levels = TRUE,
                 control = list(transform.thresh = 'min',
                                transform.latent = 'none'),
                 data = hs)

model4d <- hopit(latent.formula = latent.formula,
                 thresh.formula = ~ sex + cont_var,
                 decreasing.levels = TRUE,
                 control = list(transform.thresh = 'scale_01',
                                transform.latent = 'none'),
                 data = hs)
```

```
model4e <- hopit(latent.formula = latent.formula,
                 thresh.formula = ~ sex + cont_var,
                 decreasing.levels = TRUE,
                 control = list(transform.thresh = 'standardize',
                                transform.latent = 'none'),
                 data = hs)

model4f <- hopit(latent.formula = latent.formula,
                 thresh.formula = ~ sex + cont_var,
                 decreasing.levels = TRUE,
                 control = list(transform.thresh = 'standardize_trunc',
                                transform.latent = 'none'),
                 data = hs)

round(t(rbind(coef(model4b),
              coef(model4c),
              coef(model4d),
              coef(model4e),
              coef(model4f))),4)
```

---

`hopit.control` *Auxiliary for controlling the fitting of a* `hopit` *model*

---

## Description

An auxiliary function for controlling the fitting of a `hopit` model. Use this function to set the 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 = "BFGS",
  nlm.fit = FALSE,
  trace = TRUE,
  transform.latent = "none",
  transform.thresh = "none"
)
```

## Arguments

| | |
|---|---|
| grad.eps | an epsilon parameter ("a very small number") used to calculate the Hessian from the gradient function. |
| bgfs.maxit, cg.maxit, nlm.maxit | the maximum number of iterations. See optim and nlm for details. |
| bgfs.reltol, cg.reltol | the relative convergence tolerances for the BFGS and the CG methods. See optim for details. |
| nlm.gradtol, nlm.steptol | a tolerance at which the scaled gradient is considered close enough to zero and a minimum allowable relative step length for the nlm method. See nlm. |
| fit.methods | "CG", "BFGS", or both. If both, the CG is run first, followed by the BFGS. See optim. |
| nlm.fit | a logical; if FALSE (default) the nlm optimization method is omitted and only the BFGS and/or the CG methods are run. |
| trace | a logical for whether to trace the process of model fitting. |
| transform.latent, transform.thresh | a type of transformation applied to the all of the latent's or all of the threshold's numeric variables. Possible values:<br>• "none" : no transformation<br>• "min" : subtract the minimum from a variable<br>• "scale_01" : transform the variable to fit the range from 0 to 1<br>• "standardize" or "standardise" : subtract the mean from a variable then divide it by its standard deviation<br>• "standardize_trunc" or "standardise_trunc" : subtract the minimum from a variable then divide it by its standard deviation |

## Author(s)

Maciej J. Danko

## See Also

hopit

---

# latentIndex *Calculate the latent index*

---

## Description

Calculate the latent index from the fitted model. The latent index is a standardized latent measure that takes values from 0 to 1, where 0 refers to the worst predicted state (the maximal observed value for the latent measure) and 1 refers to the best predicted state (the minimal observed value for the latent measure).

## Usage

```
latentIndex(model, subset = NULL)

healthIndex(model, subset = NULL)
```

## Arguments

| | |
|---|---|
| model | a fitted hopit model. |
| subset | an optional vector that specifies a subset of observations. |

## Value

a vector with a latent index for each individual.

## Author(s)

Maciej J. Danko

## References

Jurges H (2007). "True health vs response styles: exploring cross-country differences in self-reported health." *Health Economics*, **16**(2), 163-178. doi:10.1002/hec.1134.

Oksuzyan A, Danko MJ, Caputo J, Jasilionis D, Shkolnikov VM (2019). "Is the story about sensitive women and stoical men true? Gender differences in health after adjustment for reporting behavior." *Social Science & Medicine*, **228**, 41-50. doi:10.1016/j.socscimed.2019.03.002.

## See Also

standardizeCoef, getCutPoints, getLevels, hopit.

## Examples

```
# DATA
data(healthsurvey)

# the order of response levels decreases from the best health to
# the worst health; hence the hopit() parameter decreasing.levels
# is set to TRUE
levels(healthsurvey$health)

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

# fit a model
model1 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                  heart_attack_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)

# calculate the health index
hi <- latentIndex(model1)

summary(hi)

# plot a simple histogram of the function output
hist(hi, col='deepskyblue3')

#plot the reported health status versus the health index.
plot(hi, response = "data", ylab = 'Health index',
     col='deepskyblue3', main = 'Reported health levels')

# plot the model-predicted health levels versus the health index.
plot(hi, response = "fitted", ylab = 'Health index',
     col='deepskyblue3', main = 'Model-predicted health levels')
```

---

`percentile_CI` *Calculating the confidence intervals of the bootstrapped function using the percentile method*

---

## Description

Calculate the confidence intervals of the bootstrapped function using the percentile method.

## Usage

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

## Arguments

| | |
|---|---|
| `boot` | a matrix or a list of vectors with bootstrapped elements. If it is list, then each element of the list is one replication. |
| `alpha` | a significance level. |
| `bounds` | which bounds to return; one of `"both"`, `"lo"`, `"up"`. |

## Author(s)

Maciej J. Danko

## See Also

`boot_hopit`, `getLevels`, `getCutPoints`, `latentIndex`, `standardiseCoef`, `hopit`.

## Examples

```
# see examples in boot_hopit() function.
```

`standardizeCoef` *Standardization of the coefficients*

## Description

Calculate standardized the coefficients (e.g. disability weights for the health variables) using the predicted latent measure obtained from the model.
In the self-rated health example the standardized coefficients are called disability weights Jurges (2007) and are calculated for each health variable to provide information about the impact of a specific health measure on the latent index (see `latentIndex`). The disability weight for a health variable is equal to the ratio of the corresponding health coefficient and the difference between the lowest and the highest values of the predicted latent health. In other words, the disability weight reduces the latent index by some given amount or percentage (i.e., the latent index of every individual is reduced by the same amount if the person had a heart attack or other heart problems)(Jurges 2007).

## Usage

```
standardizeCoef(model, namesf = identity)

standardiseCoef(model, namesf = identity)

disabilityWeights(model, namesf = identity)
```

## Arguments

| | |
|---|---|
| `model` | a fitted `hopit` model. |
| `namesf` | a vector of the names of coefficients or one argument function that modifies the names of coefficients. |

## Value

a vector with standardized coefficients.

## Author(s)

Maciej J. Danko

## References

Jurges H (2007). "True health vs response styles: exploring cross-country differences in self-reported health." *Health Economics*, **16**(2), 163-178. doi:10.1002/hec.1134.

Oksuzyan A, Danko MJ, Caputo J, Jasilionis D, Shkolnikov VM (2019). "Is the story about sensitive women and stoical men true? Gender differences in health after adjustment for reporting behavior." *Social Science & Medicine*, **228**, 41-50. doi:10.1016/j.socscimed.2019.03.002.

## See Also

latentIndex, getCutPoints, getLevels, hopit.

## Examples

```
# DATA
data(healthsurvey)

# the order of response levels decreases from the best health to
# the worst health; hence the hopit() parameter decreasing.levels
# is set to TRUE
levels(healthsurvey$health)

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

# fit a model
model1 <- hopit(latent.formula = health ~ hypertension + high_cholesterol +
                heart_attack_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)

# a function that modifies the coefficient names.
txtfun <- function(x) gsub('_',' ',substr(x,1,nchar(x)-3))

# calculate and plot the disability weights
sc <- standardizeCoef(model1, namesf = txtfun)
sc

summary(sc)

plot(sc)
```

---

`svy.varcoef_hopit` *Calculation of the variance-covariance matrix for a specified survey design (experimental function)*

---

## Description

This function is an equivalent of `survey:::svy.varcoef`. In the original approach `estfun` is calculated from glm's working residuals:
`estfun <- model.matrix(glm.object) * resid(glm.object, "working") * glm.object$weights`
In the hopit package, estfun is directly calculated as a gradient (vector of partial derivatives) of the log likelihood function. Depending on detected design an appropriate `survey` function is called.

## Usage

```
svy.varcoef_hopit(vcovMat, estfun, design)
```

## Arguments

| | |
|---|---|
| vcovMat | a variance-covariance matrix. |
| estfun | a gradient function of the log-likelihood function. |
| design | a survey.design object. |

## See Also

svydesign hopit

# Index