Package ‘rineq’

April 18, 2023

Title Concentration Index and Decomposition for Health Inequalities

Version 0.2.3

Date 2023-04-16

Description Relative, generalized, and Erreygers corrected concentration index; plot Lorenz curves; and decompose health inequalities into contributing factors. The package currently works with (generalized) linear models, survival models, complex survey models, and marginal effects probit models. originally forked by Brecht Devleesschauwer from the 'decomp' package (no longer on CRAN), rineq is now maintained by Kaspar Walter Meili. Compared to the earlier 'rineq' version on 'github' by Brecht Devleesschauwer (<https://github.com/brechtdv/rineq>), the regression tree functionality has been removed.
Improvements compared to earlier versions include improved plotting of decomposition and concentration, added functionality to calculate the concentration index with different methods, calculation of robust standard errors, and support for the decomposition analysis using marginal effects probit regression models. The development version is available at <https://github.com/kdevkdev/rineq>.

Depends R (>= 3.5.0)

Imports stats, graphics

Suggests sandwich, lmtest, mfx, survey, survival

License GPL (>= 2)

LazyData true

RoxygenNote 7.2.3

Encoding UTF-8

Language en-Us

URL https://github.com/kdevkdev/rineq/

NeedsCompilation no

Author Brecht Devleesschauwer [aut, cph],
Saveria Willimès [aut, cph],
Carine Van Malderen [aut, cph],
Peter Konings [aut, cph],
Niko Speybroeck [aut, cph],
Kaspar Meili [aut, cre, cph] (<https://orcid.org/0000-0002-9889-4406>)

Maintainer Kaspar Meili <meilikaspar@yahoo.de>

Repository CRAN

Date/Publication 2023-04-18 14:10:02 UTC

R topics documented:

---

ci *Calculates different type of concentration indexes*

Description

This function calculates the relative concentration index (Kakwani et al.), the generalized concentration index (Clarke et al., 2002), the Wagstaff index for bounded variables (Owen et al. 2016), and the concentration index with Erreygers' correction (Erreygers et al., 2009). It returns an object of class hci for which confidence intervals, summaries and plots are defined.

Usage

```
ci(
  inequar,
  outcome,
  weights = NULL,
  type = c("CI", "CIg", "CIc", "CIw"),
 method = c("linreg_delta", "linreg_convenience", "cov_convenience", "direct"),
  df_correction = TRUE,
  robust_se = FALSE,
  rse_type = "HC3"
)
```

Arguments

`ineqvar`	Used for ranking, usually relates to the socioeconomic position, for example income.
`outcome`	The variable in which the inequality should be measures, for example health.
`weights`	Optional, used to weigh the observations. Defaults to equal weights for all observations.
`type`	Character, the type of concentration index to be calculated: relative concentration index (`CI`, default), generalized concentration index (`CIg`), concentration index with Erreygers Correction `CIc`, or Wagstaff concentration index suitable for bounded and binary outcomes `CIw`
`method`	Character, defines the calculation method. One of: • `linreg_delta`: Based on linear regression without transforming the left hand side variable. Computes correct standard errors that take into account the sampling variability of the estimate of the mean (O'Donnell et al. 2008, Owen et al. 2016) • `linreg_convenience`): Based on simpler regression with transformed left hand side variable. Standard errors do not take into account thee sampling variability of the estimate of the mean(O'Donnell et al. 2008, Owen et al. 2016) • `cov_convenience`: Based on covariance. Equivalent to `linreg_convenience` (O'Donnell et al. 2008, Owen et al. 2016) • `direct`: Using direct formula, standard errors do no take weighting appropriately into account (O'Donnell et al. 2008, Kakwani et al. 1997)
`df_correction`	If `TRUE` (default), calculates the concentration index based on the population variance (derived from the sample variance).
`robust_se`	Uses robust standard errors if `TRUE`. Only available for the `linreg_*` type methods. Requires the `sandwich` package.
`rse_type`	Character, type argument for the `vcovHC()`. `HC3`' is suggested as default, set to `HC1` for Stata compatibility. See `?sandwich::vcovHC()` for options.

Value

An S3 object of class `hci`. Contains:

- `concentration_index` The concentration index
- `type` The type
- `method` The method used for calculation
- `variance` The variance,used for calculation of confidence intervals
- `fractional_rank` Computed fractional rank `NA`
- `outcome` Outcome after removing `NA`
- `call` Call signature
- `n` Number of observations after removing `NA`
- `robust_se` Were robust standard errors calculated?
- `rse_type` Type of robust standard errors.
- `df_correction` Do the degrees of freedom correspond to a sample?

References

Clarke, P. M., Gerdtham, U. G., Johannesson, M., Bingefors, K., & Smith, L. (2002). On the measurement of relative and absolute income-related health inequality. Social Science & Medicine, 55(11), 1923-1928

Erreygers, G. (2009). Correcting the concentration index. Journal of health economics, 28(2), 504-515

Kakwani, N., Wagstaff, A., & Van Doorslaer, E. (1997). Socioeconomic inequalities in health: measurement, computation, and statistical inference. Journal of econometrics, 77(1), 87-103.

O'Donnel, O., O'Neill S., Van Ourti T., & Walsh B. (2016). Conindex: Estimation of Concentration Indices. The Stata Journal 16(1): 112-138.

O'Donnell, O., Van Doorslaer, E. , Wagstaff, A., Lindelow, M., 2008. Analyzing Health Equity Using Household Survey Data: A Guide to Techniques and Their Implementation, World Bank Publications. The World Bank.

Examples

```
# Direct
data(housing)
ci.bmi <- ci(ineqvar = housing$income, outcome = housing$bmi, method = "direct")
summary(ci.bmi)

# retrieve value
ci.bmi$concentration_index

# obtain confidence intervals
confint(ci.bmi, level = 0.95)

plot(ci.bmi)

# Wagstaff type with binary outcome and robust standard errors
# that should correspond to Stata (depends on 'sandwich'):


ci.bmi.b <- ci(housing$income, housing$high.bmi, type = "CIw", robust_se = TRUE,
   rse_type = "HC1")
```

`confint.hci` *Confidence intervals for* `hci` *objects*

Description

Confidence intervals for `hci` objects

Usage

```
## S3 method for class 'hci'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`object`	An object of class `hci`
`parm`	Unused
`level`	Confidence interval level defaults to `0.95`
`...`	Unused

Value

A confidence interval in a numeric vector of length 2

Examples

```
data(housing)
ci.bmi <- ci(ineqvar = housing$income, outcome = housing$bmi, method = "direct")
confint(ci.bmi)
```

`contribution` *Function to decompose the Relative Concentration Index into its components*

Description

Currently compatible with `lm`, `glm` logit and probit, `svyglm`, `coxph` and `mfx` marginal effects probit.

Usage

```
contribution(object, ranker, correction = TRUE, type = "CI", intercept = "exclude")
```

Arguments

`object`	The model result object. class `coxph`, `glm`, `lm` or `svyglm`, `probitmfx`, `logitmfx`; the outcome should be the health variable and the predictors the components.
`ranker`	Ranking variable with the same length as the outcome.
`correction`	A logical indicating whether the global and partial confidence should be corrected for negative values using imputation.
`type`	Character, concentration index type that the decomposition should be applied to. Defaults to `CI`. Use `CIw` for binary outcomes.
`intercept`	Character, one of `exclude` or `include`, defaults to `exclude`. If `exclude`, the intercept coefficient will not included in the decomposition analysis, if set to `include`, it will be included.

Details

These functions decompose the Relative Concentration Index into its components using a (generalized) linear model, optionally using a survey design, or a Cox Proportional Hazards model. Print, summary and plot methods have been defined for the results.

If `correction` is `TRUE` negative values of components or outcome are corrected using `correct_sign()` with option `shift = FALSE`.

For non-linear models the decomposition needs to rely on a linear approximations of the effects. There are different approaches. One is to work on the scale of the `glm` coefficients and calculate the concentration index based on the predicted outcome. (Konings et al., 2010, Speybroeck et al., 2010). Another approach is to use marginal effects as beta coefficients and the original outcome (O'Donnel et al. 2008).

This function supports both. For `glm`, `coxph`, and `svyglm` models, the first approach is used. The second approach is implemented for model objects of type `probitmfx` and `logitmfx` from the 'mfx' package. See examples.

Per default, the intercept in models is excluded, but this can be changed by setting the the `intercept` argument to `include`, but this may conceptually make less sense and is more appropriate if the model does not contain an intercept.

Use `decomposition()` function directly to manually specify coefficients, outcomes, and model matrices for arbitrary models.

Value

An object of class `decomposition` containing the following components:

- `betas` A numeric vector containing regression coefficients
- `partial_cis` A numeric vector containing partial confidence intervals
- `confints` A numeric vector containing 95\
- `averages` Weighted averages of every variable in the model
- `ci_contribution` Confidence intervals for contributions
- `overall_ci` Confidence intervals for the concentration index
- `corrected_coefficients` Corrected coefficients using `correct_sign()` if, requested `FALSE` otherwise
- `outcome_corrected` Corrected outcome `correct_sign()` if requested, `FALSE` otherwise
- `rows` Rownames of used rows in the model

Warning

`ranker` should be chosen with care. Ideally, it is a variable from the same dataframe as the other variables. If not, redefine the row names in the model.

Author(s)

Peter Konings

References

Konings, P., Harper, S., Lynch, J., Hosseinpoor, A.R., Berkvens, D., Lorant, V., Geckova, A., Speybroeck, N., 2010. Analysis of socioeconomic health inequalities using the concentration index. Int J Public Health 55, 71–74. https://doi.org/10.1007/s00038-009-0078-y

Speybroeck, N., Konings, P., Lynch, J., Harper, S., Berkvens, D., Lorant, V., Geckova, A., Hosseinpoor, A.R., 2010. Decomposing socioeconomic health inequalities. Int J Public Health 55, 347–351. https://doi.org/10.1007/s00038-009-0105-z

O'Donnell, O., Doorslaer, E. van, Wagstaff, A., Lindelow, M., 2008. Analyzing Health Equity Using Household Survey Data: A Guide to Techniques and Their Implementation, World Bank Publications. The World Bank.

Examples

```
data(housing)

## Linear regression direct decomposition
fit.lm <- lm(bmi ~ sex + tenure + place + age,data = housing)

# decompose relative concentration index
contrib.lm <- contribution(fit.lm, housing$income)
summary(contrib.lm)
plot(contrib.lm, decreasing = FALSE, horiz = TRUE)


# GLM: Decomposition based on predicted outcome
fit.logit <-glm(high.bmi ~ sex + tenure + place + age, data = housing)

contrib.logit <- contribution(fit.logit, housing$income)
summary(contrib.logit)
plot(contrib.logit, decreasing = FALSE,horiz = TRUE)


# GLM probit: Decomposition based on predicted outcome
fit.probit <-glm(high.bmi ~ sex + tenure + place + age, data = housing,
                family = binomial(link = probit))

# binary, set type to 'CIw'
contrib.probit <- contribution(fit.probit, housing$income, type = "CIw")
summary(contrib.probit)
plot(contrib.probit, decreasing = FALSE,horiz = TRUE)


# Marginal effects probit using package 'mfx': Decomposition based on predicted outcome
fit.mfx <-mfx::probitmfx(high.bmi ~ sex + tenure + place + age, data = housing)

contrib.mfx <- contribution(fit.mfx, housing$income, type = "CIw")
summary(contrib.mfx, type="CIw")
plot(contrib.mfx, decreasing = FALSE, horiz = TRUE)


# package survey svy
```

```
des = survey::svydesign(~1, data= housing, weights = rep(1, NROW(housing)))
fit.svy = survey::svyglm(bmi ~ tenure+height+weight, design = des)
contrib.svy = contribution(fit.svy, housing$income)


# adopted from the `coxph` example in survival package
testcph <- data.frame(time = c(4,3,1,1,2,2,3),
              status = c(1,1,1,0,1,1,0),
              x      = c(0,2,1,1,1,0,0),
              sex    = c(0,0,0,0,1,1,1),
              income = c(100,50, 20, 20, 50, 60,100))

# Fit a stratified model
fit.coxph = survival::coxph(survival::Surv(time, status) ~ x + survival::strata(sex), testcph)
contrib.coxph = contribution(fit.coxph, testcph$income)
```

`correct_sign` *Corrects negative values in the health variable*

Description

The Relative Concentration Index is not bonded between $[-1, 1]$ if the health variable contains both negative and positive values. This function corrects for this either by imputing a value of 0 for all negative values or by subtracting the minimum value.

Usage

```
correct_sign(x, shift = TRUE)

corrected_value(x)

is_corrected(x)
```

Arguments

`x`	A numeric vector, typically representing health.
`shift`	If `FALSE` (the default), 0 is imputed for all negative values in `x`. If `TRUE` the minimum value of `x` is subtracted from it.

Value

`correct_sign()` returns a list with 2 components:

- correctedcorrected version of `x`
- modifiedlogical, `TRUE` when any of the elements of `x` have been changed `corrected_value()` returns the corrected value if passed the result of `correct_sign()` `is_corrected()` returns `TRUE` if a modifications was made if passed the result of `correct_sign()`, `FALSE` otherwise

These components can be extracted with the functions `corrected_value` and `is_corrected`.

Functions

- `corrected_value()`: Return the corrected value
- `is_corrected()`: Check if the sign was corrected

Author(s)

Peter Konings

Examples

```
data("housing")

# standardize & normalize bmi, will introduce negative values
housing$bmi.std <- (housing$bmi - mean(housing$bmi))/ sd(housing$bmi)

housing$bmi.std.shifted <- corrected_value(correct_sign(housing$bmi.std, shift = TRUE))
housing$bmi.std.imputed <- corrected_value(correct_sign(housing$bmi.std, shift = FALSE))

## compare the effect of both methods
plot(density(housing$bmi.std, na.rm = TRUE))
points(density(housing$bmi.std.shifted, na.rm = TRUE), col = 'blue')
points(density(housing$bmi.std.imputed, na.rm = TRUE), col = 'green')
```

`decomposition` *Decomposition analysis*

Description

Used by the wrapper `contribution()` but can be used manually. Calculates the decomposition for a given regression model.

Usage

```
decomposition(outcome, betas, mm, ranker, wt, correction, citype = "CI")
```

Arguments

`outcome`	Outcome variable
`betas`	Beta coefficients from regression.
`mm`	Model matrix from regression
`ranker`	Ranking variable
`wt`	Weights
`correction`	Apply sign correction?
`citype`	Character, CI type to be calculated, defaults to `CI`. Use `CIw` for binary outcomes.

Value

S3 object of class decomposition

Examples

```
fit.lm = lm(mtcars$mpg ~ mtcars$cyl)
decomp = decomposition(mtcars$mpg, coefficients(fit.lm), fit.lm$model,
                       mtcars$hp, wt = rep(1, nrow(mtcars)), correction = FALSE)
summary(decomp)
```

`housing` *Artificial example data on housing conditions*

Description

Microdata with a permissive license that includes continuous data on health and income is hard to come by. In stead of real data, the package thus includes an imaginary dataset.

Usage

```
data(housing)
```

Format

`data.frame` object.

Variable list:

id unique identifier per person

sex female or male

age integer, from 20 to 94

tenure categorical. One of homeless, irregular, own_apartment, own_house, or rent

height height in cm

weight weight in kg

bmi weight/(height/100)^2

income continuous, imaginary currency without unit

Source

Artificially generated by the package authors

`plot.decomposition`	*Plots a barplot of the contribution percentages in a* `decomposition` *object. Sets custom plot margins and uses the graphical parameters* `xlim`*,* `horiz`*,* `las` *and* `xlab` *which therefore cannot be customized*

Description

Plots a barplot of the contribution percentages in a `decomposition` object. Sets custom plot margins and uses the graphical parameters `xlim`, `horiz`, `las` and `xlab` which therefore cannot be customized

Usage

```
## S3 method for class 'decomposition'
plot(x, decreasing = TRUE, horiz = FALSE, ...)
```

Arguments

`x`	Object returned from decomposition function
`decreasing`	Whether to sort contributions decreasing or not
`horiz`	If the barplots should be printed horizontally or vertically
`...`	Graphical parameter passed on to `base::barplot()`

Value

Invisibly returns `x` as the function is called for side effects (plotting).

Examples

```
data(housing)
# Linear regression & decompose
fit.lm <- lm(bmi ~ sex + tenure + place + age,data = housing)
contrib.lm <- contribution(fit.lm, housing$income)

# plot horizontally, in increasing order
plot(contrib.lm, decreasing = FALSE, horiz = TRUE)
```

`plot.hci`	*Plots the concentration curve for an* `hci` *object.*

Description

Plots the concentration curve for an `hci` object.

Usage

```
## S3 method for class 'hci'
plot(x, ...)
```

Arguments

`x`	Object with of `hci`
`...`	Further arguments passed to `base::plot()`

Value

Invisibly returns `x` as the function is called for side effects (plotting).

Examples

```
data(housing)
ci.bmi <- ci(ineqvar = housing$income, outcome = housing$bmi, method = "direct")
plot(ci.bmi)
```

`print.decomposition` *Print function for* `decomposition` *objects.*

Description

Print function for `decomposition` objects.

Usage

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

Arguments

`x`	Object of type `decomposition`
`...`	Currently unused

Value

Invisibly returns `x` as the function is called for side effects.

Examples

```
data(housing)
# Linear regression &  decompose
fit.lm <- lm(bmi ~ sex + tenure + place + age,data = housing)
contrib.lm <- contribution(fit.lm, housing$income)

# print
print(contrib.lm)
```

`print.hci` *Prints an* `hci` *object.*

Description

Prints an `hci` object.

Usage

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

Arguments

`x`	Object of type `hci`
`...`	Currently unused

Value

Invisibly returns `x` as the function is called for side effects.

Examples

```
data(housing)
ci.bmi <- ci(ineqvar = housing$income, outcome = housing$bmi, method = "direct")
print(ci.bmi)
```

rank_wt *Calculates the weighted rank*

Description

Calculates the weighted rank

Usage

```
rank_wt(x, wt)
```

Arguments

`x`	numeric vector
`wt`	weights

Value

A numeric vector containing weighted fractional ranks of the elements of `x`.

Author(s)

Peter Konings

References

Kakwani *et al.*, 1997.

Examples

```
x <- sample(1:10, size = 10, replace = TRUE)
x.weight <- seq(0, 1, length.out = 10)
rank_wt(x, wt = x.weight)
```

summary.decomposition *Prints and returns a summary for a* `decomposition` *object.*

Description

Prints and returns a summary for a `decomposition` object.

Usage

```
## S3 method for class 'decomposition'
summary(object, digits = getOption("digits"), addcoefs = FALSE, ...)
```

Arguments

`object`	Result of a decomposition analysis, of class `decomposition`
`digits`	Number of digits, defaults to R `digits` option
`addcoefs`	Whether or not to add coefficients (defaults to `FALSE`)
`...`	Additional parameters, currently unused

Value

A data frame frame with columns for the absolute and relative contribution, elasticity, concentration index including confidence intervals, and whether correction was applied. If specified using `addcoefs`, the coefficients are included as the first column.

Examples

```
data(housing)
# Linear regression & decompose
fit.lm <- lm(bmi ~ sex + tenure + place + age,data = housing)
contrib.lm <- contribution(fit.lm, housing$income)

# print
print(contrib.lm)
```

`summary.hci` *Prints the a summary of the concentration index object* `hci`

Description

Prints the a summary of the concentration index object `hci`

Usage

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

Arguments

`object`	Object of type `hci`
`...`	Currently unused

Value

No returns value. Directly prints to the standard output connection.

Examples

```
data(housing)
ci.bmi <- ci(ineqvar = housing$income, outcome = housing$bmi, method = "direct")
summary(ci.bmi)
```

`var_wt` *Calculates the weighted variance*

Description

Calculates the weighted variance

Usage

```
var_wt(x, wt, na.rm = FALSE)
```

Arguments

`x`	numeric vector
`wt`	weights
`na.rm`	If `TRUE`, indices where `x` is NA will be removed

Value

A numeric vector containing weighted variance of the elements of `x`

Examples

```
x <- sample(1:10, size = 10, replace = TRUE)
x.weight <- seq(0, 1, length.out = 10)
var_wt(x, wt = x.weight)
```

Index