Package 'binequality'

February 18, 2018

Type Package

Title Methods for Analyzing Binned Income Data

Version 1.0.3

Date 2018-02-18

Author Samuel V. Scarpino, Paul von Hippel, and Igor Holas

Maintainer Samuel V. Scarpino <s.scarpino@northeastern.edu>

Description Methods for model selection, model averaging, and calculating metrics, such as the Gini, Theil, Mean Log Deviation, etc, on binned income data where the topmost bin is right-censored. We provide both a non-parametric method, termed the bounded midpoint estimator (BME), which assigns cases to their bin midpoints; except for the censored bins, where cases are assigned to an income estimated by fitting a Pareto distribution. Because the usual Pareto estimate can be inaccurate or undefined, especially in small samples, we implement a bounded Pareto estimate that yields much better results. We also provide a parametric approach, which fits distributions from the generalized beta (GB) family. Because some GB distributions can have poor fit or undefined estimates, we fit 10 GB-family distributions and use multimodel inference to obtain definite estimates from the best-fitting distributions. We also provide binned income data from all United States of America school districts, counties, and states.

License GPL (>= 3.0)

LazyLoad yes

Depends R (>= 2.10), gamlss (>= 4.2.7), gamlss.cens (>= 4.2.7), gamlss.dist (>= 4.3.0)

Imports survival (>= 2.37-7), ineq (>= 0.2-11)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-18 21:05:06 UTC

R topics documented:

`binequality-package` *Methods for Analyzing Binned Income Data*

Description

Methods for model selection, model averaging, and calculating metrics, such as the Gini, Theil, Mean Log Deviation, etc, on binned income data where the topmost bin is right-censored. We provide both a non-parametric method, termed the bounded midpoint estimator (BME), which assigns cases to their bin midpoints; except for the censored bins, where cases are assigned to an income estimated by fitting a Pareto distribution. Because the usual Pareto estimate can be inaccurate or undefined, especially in small samples, we implement a bounded Pareto estimate that yields much better results. We also provide a parametric approach, which fits distributions from the generalized beta (GB) family. Because some GB distributions can have poor fit or undefined estimates, we fit 10 GB-family distributions and use multimodel inference to obtain definite estimates from the best-fitting distributions. We also provide binned income data from all United States of America school districts, counties, and states.

Details

Package:	binequality
Type:	Package
Version:	1.0.3
Date:	2018-02-18
License:	GPL (>= 3.0)

The datasets are: state_bins, county_bins, and school_district_bins.

The main functions are: fitFunc, run_GB_family, getMids, getQuantilesParams, giniCoef, LRT, makeFitComb, makeInt, makeIntWeight, makeWeightsAIC, mAvg, midStats, MLD, modelAvg, paramFilt, and theilInd.

Type ?<object> to learn more about these objects, e.g. ?state_bins

Type ?<function> to see examples of the function's use, e.g. ?getMids

Author(s)

Samuel V. Scarpino, Paul von Hippel, and Igor Holas

Maintainer: Samuel V. Scarpino <s.scarpino@northeastern.edu>

References

Von Hippel, P. T., Scarpino, S. V., & Holas, I. (2016). Robust estimation of inequality from binned incomes. Sociological Methodology, 46(1), 212-251.

See Also

gamlss

Examples

```
#FIXME, write the example run of states here
```

---

county_bins *A data set containing binned income for US counties*

Description

A data set of binned income by US counties. FIXME - more info.

Usage

```
data(state_bins)
```

Format

#FIXME add in format

Examples

```
data(county_bins)
```

fitFunc *A function to fit a parametric distribution to binned data.*

Description

This function fits a parametric distribution binned data. The data are subdivided using ID.

Usage

```
fitFunc(ID, hb, bin_min, bin_max, obs_mean, ID_name,
  distribution = "LOGNO", distName = "LNO", links = c(muLink =
  "identity", sigmaLink = "log", nuLink = NULL, tauLink = NULL),
   qFunc = qLOGNO, quantiles = seq(0.006, 0.996, length.out =
   1000), linksq = c(identity, exp, NULL, NULL), con =
   gamlss.control(c.crit=0.1,n.cyc=200, trace=FALSE),
   saveQuants = FALSE, muStart = NULL, sigmaStart = NULL,
    nuStart = NULL, tauStart = NULL, muFix = FALSE,
    sigmaFix = FALSE, nuFix = FALSE, tauFix = FALSE,
    freeParams = c(TRUE, TRUE, FALSE, FALSE),
    smartStart = FALSE, tstamp = as.numeric(Sys.time()))
```

Arguments

ID	a (non-empty) object containing the group ID for each row. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
hb	a (non-empty) object containing the number of observations in each bin. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
bin_min	a (non-empty) object containing the lower bound of each bin. Currently, this package cannot handle data with open lower bounds. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
bin_max	a (non-empty) object the upper bound of each bin. Currently, this package can only handle the upper-most bin being open ended. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
obs_mean	a (non-empty) object containing the mean for each group. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
ID_name	a (non-empty) object containing column name for the ID column.
distribution	a (non-empty) character naming a gamlss family.
distName	a (non-empty) character object with the name of the distribution.
links	a (non-empty) vector of link characters naming functions with the following items: muLink, sigmaLink, nuLink, and tauLink.
qFunc	a (non-empty)gamlss function for calculating quantiles, this should match the distribution in distribution.

quantiles	a (non-empty) numeric vectors of the desired quantiles, these are used in calculating metrics.
linksq	a (non-empty) vector of functions, which undue the link functions. For example, if muLink = log, then the first entry in linksq should be exp. If you are using an indentity link function in links, then the corresponding entry in linksq should be indentity.
con	an optional lists modifying gamlss.control.
saveQuants	an optional logical value indicating whether to save the quantiles.
muStart	an optional numerical value for the starting value of mu.
sigmaStart	an optional numerical value for the starting value of sigma.
nuStart	an optional numerical value for the starting value of nu.
tauStart	an optional numerical value for the starting value of tau.
muFix	an logical value indicating whether mu is fixed or is free to vary during the fitting process.
sigmaFix	an logical value indicating whether sigma is fixed or is free to vary during the fitting process.
nuFix	an logical value indicating whether nu is fixed or is free to vary during the fitting process.
tauFix	an logical value indicating whether tau is fixed or is free to vary during the fitting process.
freeParams	a vector of logical values indicating whether each of the four parameters is free == TRUE or fixed == FALSE.
smartStart	a logical indicating whether a smart starting place should be chosen, this applies only when fitting the GB2 distribution.
tstamp	a time stamp.

Details

Fits a GAMLSS and estimates a number of metrics, see value.

Value

returns a list with 'datOut' a data.frame with the IDs, observer mean, distribution, estimated mean, variance, coefficient of variation, cv squared, gini, theil, MLD, aic, bic, the results of a convergence test, log likelihood, number of parameters, median, and std. deviation; 'timeStamp' a time stamp; 'parameters' the estiamted parameter; and 'quantiles' the quantile estimates if saveQuants == TRUE)

References

FIXME - references

See Also

gamlss

Examples

```
data(state_bins)

use_states <- which(state_bins[,'State'] == 'Texas' | state_bins[,'State'] == 'California')

ID <- state_bins[use_states,'State']
hb <- state_bins[use_states,'hb']
bmin <- state_bins[use_states,'bin_min']
bmax <- state_bins[use_states,'bin_max']
omu <- rep(NA, length(use_states))
fitFunc(ID = ID, hb = hb, bin_min = bmin, bin_max = bmax, obs_mean = omu, ID_name = 'State')
```

---

getMids *A function to calculate the bin midpoints.*

Description

This function returns the bin midpoints for use in calculating midpoint-based statistics. The code is not written to handle left censored bins or more than one right censored bins.

Usage

```
getMids(ID, hb, lb, ub, alpha_bound)
```

Arguments

ID	A vector of group IDs
hb	A vector of heights for each bin
lb	A vector of lower bounds, which must all be real numbers.
ub	A vector of upper bouns, which can have one censored bin per group.
alpha_bound	A numeric value indicating the bound on alpha for determining the midpoint of the upper-most, censored bin. To unbound the value of alpha, set alpha_bound = numeric(0).

Details

For all non-censored bins, ie real number upper and lower bounds, the midpoint is simply the arithemtic mean of the bounds. This assumes the number of observations are normally distributed in each bin. However, for the right-censored bin we relax this assumption when calcuting the bin midpoint. FIXME - say something about the method we use to calculate midpoint of the upper bin.

Value

returns a list with the following elements:

References

FIXME - references

Examples

```
data(state_bins)

bin_mids <- getMids(ID = state_bins[,'State'], hb =
state_bins[,'hb'], lb = state_bins[,'bin_min'], ub =
state_bins[,'bin_max'], alpha_bound = 10/9)

bin_mids_unbound <- getMids(ID = state_bins[,'State'], hb =
state_bins[,'hb'], lb = state_bins[,'bin_min'], ub =
state_bins[,'bin_max'], alpha_bound = numeric(0))
```

---

getQuantilesParams *A function to extract the quantiles and parameters*

Description

This extracts the quantiles and parameters.

Usage

```
getQuantilesParams(fit.i, qFunc = qLOGNO, quantiles = seq(0.006,
0.996, length.out = 1000), linksq = c(identity, exp, NULL,
NULL), freeParams, fixedParams)
```

Arguments

fit.i	a (non-empty) object of class gamlss, which is the fitted distribution.
qFunc	a (non-empty) quantile generating function from gamlss.
quantiles	an optional numeric vector of the desired quantiles.
linksq	a (non-empty) vector is link functions.
freeParams	a (non-empty) logical vector inidicating whether parameters are fixed == FALSE or free == TRUE.
fixedParams	a (non-empty) numeric vector of fixed parameter values.

Details

Extracts the quantile and parameter estimates.

Value

Returns a list with: samps = the quantiles extracted at the locations specified in quantiles and params = the parameter values of the fitted model.

References

FIXME - references

Examples

```
#not run, this function is used internally
```

---

giniCoef *Calculates the Gini coefficient from quantiles*

Description

This function estimates the Gini coefficient using the quantiles sampled from a fitted GAMLSS object.

Usage

```
giniCoef(seq.i, samps.i)
```

Arguments

seq.i	a (non-empty) numeric vector of quantile locations.
samps.i	a (non-empty) numeric vector of quantile values.

Details

Calculates the Gini coefficient from quantiles

Value

Returns the Gini coefficient estimate.

References

FIXME - references

Examples

```
#not run, internal function
```

LRT *A function to perform likelihood ratio tests*

Description

This function performs likelihood ratio tests for use in model selection.

Usage

```
LRT(dat, fitComb, ID)
```

Arguments

dat	a (non-empty) data frame containing an ID column (see ID) and a column called "hb", which contains the bin heights used in model fitting.
fitComb	a (non-empty) data frame returned from the function makeFitComb.
ID	a (non-empty) string indicating the column dat and the column in fitComb containing the group ids, these column names must match.

Details

Performs a likelihood ratio test.

Value

Returns an object with the same information as was passed into the fitComb argument, but with additional columns G2 = the test statistic, p = the p value, and df = the degrees of freedom.

References

FIXME - references

Examples

```
#not run, function is used internally
```

makeFitComb *A function to transform a list into a dataframe*

Description

This funciton transforms a list of fitted models into a data frame.

Usage

```
makeFitComb(distFitsList)
```

Arguments

distFitsList	a (non-empty) list containing the fitted models from the function fitFunc

Details

Transforms a list into a data frame

Value

Returns a data frame where each row is a distribution/unique ID pair, ie a county and its fitted GB2 distribution. It also contains summary stats.

Examples

```
#not run, internal function
```

makeInt *A function to create a survival object from bin counts.*

Description

This function creates a survival object from bin counts, these data cannot be right censored. There is a separate function for the right censored bins.

Usage

```
makeInt(ints, hb, trans = NULL)
```

Arguments

ints	a (non-empty) numeric matrix containing the bin end points, there must be as many rows as length(hb) and the lower bin should be on the left.
hb	a (non-empty) numeric vector of bin heights.
trans	an optional function to transform the bins.

Details

Creates a survival object. FIXME - reference to survival package and Surv function.

Value

Returns an object of class "Surv"

See Also

FIXME - reference to survival package and Surv function.

Examples

```
#not run, internal function
```

---

makeIntWeight *A function to create a survival object from bin counts and normalized bin weights.*

Description

This function creates a survival object from bin counts, these data cannot be right censored. There is a separate function for the right censored bins. It also returns normalized bin weights.

Usage

```
makeIntWeight(ints, hb, trans = NULL)
```

Arguments

ints	a (non-empty) numeric matrix containing the bin end points, there must be as many rows as length(hb) and the lower bin should be on the left.
hb	a (non-empty) numeric vector of bin heights.
trans	an optional function to transform the bins.

Details

Creates a survival object. FIXME - reference to survival package and Surv function.

Value

Returns an object of class "Surv" and normalizde bin weights.

See Also

FIXME - reference to survival package and Surv function.

Examples

```
#not run, internal function
```

`makeWeightsAIC` *A function to calculate AIC weights*

Description

This function calculates AIC weights for use in model selection and model averaging.

Usage

```
makeWeightsAIC(aics)
```

Arguments

`aics`	a (non-empty) numeric vector of AIC values.

Details

FIXME - citation

Value

Returns a vector of AIC weights.

References

FIXME - refs

Examples

```
#not run, internal function
```

mAvg *A simple function to perfom model averaging using pre-calculated weights.*

Description

This function takes weights and parameters and returns model averaged parameters using the weights.

Usage

```
mAvg(params, ws)
```

Arguments

params	a (non-empty) numeric vector of parameters
ws	a (non-empty) numeric vector of weights corresponding to params.

Value

Returns a numeric vector of model averaged parameters.

Examples

```
#not run, iternal function
```

midStats *A function to calculate statistics using bin midpoints*

Description

This function calcualtes a suite of pre-defined statistics on binned distributions using pre-calculated midpoints for each bin.

Usage

```
midStats(data)
```

Arguments

data	a (non-empty) data frame with columns named ID, mids, and hb.

Details

Currently has the following stats: 'mean','median','gini','theil','cv','MLD'. FIXME - reference to functions in gamlss

Value

Returns a data frame with the following columns: 'ID','mean','median','gini','theil','cv','MLD'

See Also

FIXME add in links to Gini and Theil functions.

Examples

```
#not run, internal function
```

MLD *A function to calculate the MLD*

Description

This fuction calculates MLD

Usage

```
MLD(samps)
```

Arguments

samps	a (non-empty) numeric vector of values to calculate MLD over, for example, bin mid points or samples take from a fitted distribution.

Details

FIXME - equations

Value

returns a numeric value representing the MLD

References

FIXME - references

Examples

```
MLD(qnorm(seq(0.001,0.999,length.out = 10), mean = 100))
```

`modelAvg` *A function to calculate model averages*

Description

This function calculates model averaged statistics using AIC and BIC.

Usage

```
modelAvg(fitList, ID, nonCon = TRUE)
```

Arguments

`fitList`	a (non-empty) list of fitted distributions
`ID`	a (non-empty) string of the ID column name.
`nonCon`	an optional logical, where nonCon == TRUE excludes models failing to converged and nonCon == FALSE includes them.

Details

Calculates model averaged statistics using BIC and AIC as weights.

Value

Returns a list with aic and bic values, aic and bic averages, and the aic and bic weights.

Examples

```
#not run, internal function
```

`paramFilt` *A function to filter models based on estimated parameters*

Description

This functions filters models, excludes them from model averaging, and sets all parameter values/statistics to NA. The filter determines which models should have undefined moments based on the estimated parameters. These are filtered. FIXME - add in specific criteria for filtering.

Usage

```
paramFilt(paramsList, fitComb)
```

Arguments

paramsList	a (non-empty) list containing the parameter values, it should be of length(fitComb).
fitComb	a (non-empty) list of fitted models

Value

Returns fitComb, filtered based on the parameters.

References

FIXME - references for specific filtering criteria

Examples

```
#not run, internal function
```

---

run_GB_family *A function to fit a parametric distributions to binned data.*

Description

This function fits a series of parametric distributions from the GB family to binned data.

Usage

```
run_GB_family(ID,hb,bin_min,bin_max,obs_mean,ID_name,quantiles,modelsToFit,return_params)
```

Arguments

ID	a (non-empty) object containing the group ID for each row. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
hb	a (non-empty) object containing the number of observations in each bin. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
bin_min	a (non-empty) object containing the lower bound of each bin. Currently, this package cannot handle data with open lower bounds. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
bin_max	a (non-empty) object the upper bound of each bin. Currently, this package can only handle the upper-most bin being open ended. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
obs_mean	a (non-empty) object containing the mean for each group. Importantly, ID, bh, bin_min, bin_max, and obs_mean MUST be the same length and be in the SAME order.
ID_name	a (non-empty) object containing column name for the ID column.

quantiles	a (non-empty) vector of quantiles for use in calculating metrics. The default is seq(0.006,0.996, length.out = 1000).
modelsToFit	a (non-empty) vector of model names as characters, currently limited to the following distributions GB2, GG, BETA2, DAGUM, SINGMAD, LOGNO, WEI, GA, LOGLOG, PARETO2.
return_params	a (non-empty) logical indicating whether to retunr the parameters. The default is TRUE.

Details

Fits a GAMLSS and estimates a number of metrics, see value.

Value

returns a list with "fit" = fitted model and metrics, "fit.filter" = filtered for undefined moments fitted model and metrics, "best_model" = best model results, "best_model.filter" = filtered for undefined moments best model results.

See Also

gamlss

Examples

```
data(state_bins)

use_states <- which(state_bins[,'State'] == 'Texas')
TX <- state_bins[use_states,]

LNO_WEI_GA <- run_GB_family(ID = TX[,'State'], hb = TX[,'hb'],
bin_min = TX[,'bin_min'], bin_max = TX[,'bin_max'], obs_mean =
rep(NA, length(use_states)), ID_name = "State", quantiles =
seq(0.006,0.996, length.out = 1000), modelsToFit =
c('LOGNO','WEI','GA'))
```

---

school_district_bins *A data set containing the school district data.*

Description

This is a list where each entry is a different year.

Usage

```
data(state_bins)
```

Format

#FIXME - add in format

Examples

```
data(school_district_bins)
head(school_district_bins[[1]])
names(school_district_bins)
```

SDL *A function to calculate the SDL*

Description

This fuction calculates SDL

Usage

```
SDL(samps)
```

Arguments

samps	a (non-empty) numeric vector of values to calculate SDL over, for example, bin mid points or samples take from a fitted distribution.

Details

FIXME - equations

Value

returns a numeric value representing the SLD

References

FIXME - references

Examples

```
SDL(qnorm(seq(0.001,0.999,length.out = 10), mean = 100))
```

state_bins *A data set containing the binned state data.*

Description

A data set of binned income by US counties. FIXME - more info.

Usage

```
data(state_bins)
```

Examples

```
data(state_bins)
```

theilInd *A function to calculate the Theil*

Description

This fuction calculates the Theil Index.

Usage

```
theilInd(samps)
```

Arguments

samps	a (non-empty) numeric vector of values to calculate MLD over, for example, bin mid points or samples take from a fitted distribution.

Details

FIXME - equations

Value

returns a numeric value representing the Theil Index

References

FIXME - references

Examples

```
theilInd(qnorm(seq(0.001,0.999,length.out = 10), mean = 100))
```

Index