Package ‘Kernelheaping’

October 12, 2022

Type Package
Title Kernel Density Estimation for Heaped and Rounded Data
Version 2.3.0
Date 2022-01-26
Depends R (>= 2.15.0), MASS, ks, sparr
Imports sp, plyr, dplyr, fastmatch, fitdistrplus, GB2, magrittr, mvtnorm
Author Marcus Gross [aut, cre],
Lukas Fuchs [aut],
Kerstin Erfurth [ctb]
Maintainer Marcus Gross <marcus.gross@inwt-statistics.de>
Description In self-reported or anonymised data the user often encounters heaped data, i.e. data which are rounded (to a possibly different degree of coarseness). While this is mostly a minor problem in parametric density estimation the bias can be very large for non-parametric methods such as kernel density estimation. This package implements a partly Bayesian algorithm treating the true unknown values as additional parameters and estimates the rounding parameters to give a corrected kernel density estimate. It supports various standard bandwidth selection methods. Varying rounding probabilities (depending on the true value) and asymmetric rounding is estimable as well: Gross, M. and Rendtel, U. (2016) (<doi:10.1093/jssam/smw011>).
Additionally, bivariate non-parametric density estimation for rounded data, Gross, M. et al. (2016) (<doi:10.1111/rss.a.12179>), as well as data aggregated on areas is supported.
License GPL-2 | GPL-3
RoxygenNote 7.1.0
NeedsCompilation no
Repository CRAN
Date/Publication 2022-01-26 18:42:52 UTC
createSim.Kernelheaping

Create heaped data for Simulation

Description

Create heaped data for Simulation

Usage

createSim.Kernelheaping(
  n,
  distribution,
  rounds,
  thresholds,
  offset = 0,
  downbias = 0.5,
  Beta = 0,
  ...
)

Arguments

  n          sample size
  distribution name of the distribution where random sampling is available, e.g. "norm"
  rounds     rounding values
\textit{dbivr}

- **thresholds**: rounding thresholds (for Beta=0)
- **offset**: certain value added to all observed random samples
- **downbias**: bias parameter
- **Beta**: acceleration parameter
- **...**: additional attributes handed over to "rdistribution" (i.e. rnorm, rgamma, ...)

\textbf{Value}

List of heaped values, true values and input parameters

\begin{itemize}
  \item \textit{dbivr} \quad \textit{Bivariate kernel density estimation for rounded data}
\end{itemize}

\textbf{Description}

Bivariate kernel density estimation for rounded data

\textbf{Usage}

\begin{verbatim}
  dbivr(
    xrounded,     # rounded values from which to estimate bivariate density, matrix with 2 columns (x,y)
    roundvalue,   # rounding value (side length of square in that the true value lies around the rounded one)
    burnin = 2,   # burn-in sample size
    samples = 5,  # sampling iteration size
    adaptive = FALSE,  # set to TRUE for adaptive bandwidth
    gridsize = 200  # number of evaluation grid points
  )
\end{verbatim}

\textbf{Arguments}

- **xrounded**: rounded values from which to estimate bivariate density, matrix with 2 columns (x,y)
- **roundvalue**: rounding value (side length of square in that the true value lies around the rounded one)
- **burnin**: burn-in sample size
- **samples**: sampling iteration size
- **adaptive**: set to TRUE for adaptive bandwidth
- **gridsize**: number of evaluation grid points
Value

The function returns a list object with the following objects (besides all input objects):

- **Mestimates**: kde object containing the corrected density estimate
- **gridx**: Vector Grid on which density is evaluated (x)
- **gridy**: Vector Grid on which density is evaluated (y)
- **resultDensity**: Array with Estimated Density for each iteration
- **resultX**: Matrix of true latent values X estimates
- **delaigle**: Matrix of Delaigle estimator estimates

Examples

```r
# Create Mu and Sigma -----------------------------------------------------------
mu1 <- c(0, 0)
mu2 <- c(5, 3)
mu3 <- c(-4, 1)
Sigma1 <- matrix(c(4, 3, 3, 4), 2, 2)
Sigma2 <- matrix(c(3, 0.5, 0.5, 1), 2, 2)
Sigma3 <- matrix(c(5, 4, 4, 6), 2, 2)
# Mixed Normal Distribution -------------------------------------------------------
mus <- rbind(mu1, mu2, mu3)
Sigmas <- rbind(Sigma1, Sigma2, Sigma3)
props <- c(1/3, 1/3, 1/3)
## Not run: xtrue=rmvnorm.mixt(n=1000, mus=mus, Sigmas=Sigmas, props=props)
roundvalue=2
xrounded=plyr::round_any(xtrue,roundvalue)
est <- dbivr(xrounded,roundvalue=roundvalue,burnin=5,samples=10)

#Plot corrected and Naive distribution
plot(est,trueX=xtrue)
# for comparison: plot true density
dens=dmvnorm.mixt(x=expand.grid(est$Mestimates$eval.points[[1]],est$Mestimates$eval.points[[2]]),
mus=mus, Sigmas=Sigmas, props=props)
dens=matrix(dens,nrow=length(est$gridx),ncol=length(est$gridy))
contour(dens,x=est$Mestimates$eval.points[[1]],y=est$Mestimates$eval.points[[2]],
xlim=c(min(est$gridx),max(est$gridx)),ylim=c(min(est$gridy),max(est$gridy)),main="True Density")
## End(Not run)
```

**dc**

**Kernel density estimation for classified data**

**Description**

Kernel density estimation for classified data
Usage

dclass(
  xclass,  
  burnin = 2,  
  samples = 5,  
  boundary = FALSE,  
  bw = "nrd0",  
  evalpoints = 200,  
  adjust = 1,  
  dFunc = NULL
)

Arguments

  xclass          classified values; matrix with two columns: lower and upper value
  burnin          burn-in sample size
  samples         sampling iteration size
  boundary        TRUE for positive only data (no positive density for negative values)
  bw               bandwidth selector method, defaults to "nrd0" see density for more options
  evalpoints      number of evaluation grid points
  adjust          as in density, the user can multiply the bandwidth by a certain factor such that
                   bw=adjust*bw
  dFunc           character optional density (with "d", "p" and "q" functions) function name for
                   parametric estimation such as "norm" "gamma" or "lnorm"

Value

  The function returns a list object with the following objects (besides all input objects):

    Mestimates      kde object containing the corrected density estimate
    gridx           Vector Grid on which density is evaluated
    resultDensity   Matrix with Estimated Density for each iteration
    resultX         Matrix of true latent values X estimates

Examples

  x=rlnorm(500, meanlog = 8, sdlog = 1)
  classes <- c(0,500,1000,1500,2000,2500,3000,4000,5000,6000,8000,10000,15000,Inf)
  xclass <- cut(x,breaks=classes)
  xclass <- cbind(classes[as.numeric(xclass)], classes[as.numeric(xclass) + 1])
  densityEst <- dclass(xclass=xclass, burnin=20, samples=50, evalpoints=1000)
  plot(densityEst$Mestimates~densityEst$gridx ,lwd=2, type = "l")
Kernel density estimation for heaped data

Usage

dheaping(
xheaped, rounds,
burnin = 5, samples = 10,
setBias = FALSE, weights = NULL,
bw = "nrd0", boundary = FALSE,
unequal = FALSE, random = FALSE,
adjust = 1, recall = F,
recallParams = c(1/3, 1/3)
)

Arguments

xheaped heaped values from which to estimate density of x
rounds rounding values, numeric vector of length >=1
burnin burn-in sample size
samples sampling iteration size
setBias if TRUE a rounding Bias parameter is estimated. For values above 0.5, the respondents are more prone to round down, while for values < 0.5 they are more likely to round up
weights optional numeric vector of sampling weights
bw bandwidth selector method, defaults to "nrd0" see density for more options
boundary TRUE for positive only data (no positive density for negative values)
unequal if TRUE a probit model is fitted for the rounding probabilities with log(true value) as regressor
random if TRUE a random effect probit model is fitted for rounding probabilities
adjust as in density, the user can multiply the bandwidth by a certain factor such that bw=adjust*bw
recall if TRUE a recall error is introduced to the heaping model
recallParams recall error model parameters expression(nu) and expression(eta). Default is c(1/3, 1/3)
**dheaping**

**Value**

The function returns a list object with the following objects (besides all input objects):

- **meanPostDensity**: Vector of Mean Posterior Density
- **gridx**: Vector Grid on which density is evaluated
- **resultDensity**: Matrix with Estimated Density for each iteration
- **resultRR**: Matrix with rounding probability threshold values for each iteration (on probit scale)
- **resultBias**: Vector with estimated Bias parameter for each iteration
- **resultBeta**: Vector with estimated Beta parameter for each iteration
- **resultX**: Matrix of true latent values X estimates

**Examples**

```r
# Simple Rounding
xtrue <- rnorm(3000)
xrounded <- round(xtrue)
est <- dheaping(xrounded, rounds=1, burnin=20, samples=50)
plot(est, trueX=xtrue)

# Real Data Example
# Student learning hours per week
data(students)
xheaped <- as.numeric(na.omit(students$StudyHrs))
## Not run: est <- dheaping(xheaped, rounds=c(1, 2, 5, 10), boundary=TRUE, unequal=TRUE, burnin=20, samples=50)
plot(est)
summary(est)
## End(Not run)

# Simulate Data
Sim1 <- createSim.Kernelheaping(n=500, distribution="norm", rounds=c(1, 10, 100), thresholds=c(-0.5244005, 0.5244005), sd=100)
## Not run: est <- dheaping(Sim1$xheaped, rounds=Sim1$rounds)
plot(est, trueX=Sim1$x)
summary(est)
tracePlots(est)
## End(Not run)

# Biased rounding
Sim2 <- createSim.Kernelheaping(n=500, distribution="gamma", rounds=c(1, 2, 5, 10),
thresholds=c(-1.2815516, -0.6744898, 0.3853205), downbias=0.2, shape=4, scale=8, offset=45)
## Not run: est <- dheaping(Sim2$xheaped, rounds=Sim2$rounds, setBias=T, bw="SJ")
plot(est, trueX=Sim2$x)
summary(est)
tracePlots(est)
## End(Not run)
```
Sim3 <- createSim.Kernelheaping(n=500, distribution="gamma", rounds=c(1,2,5,10),
thresholds=c(1.84, 2.64, 3.05), downbias=0.75, Beta=-0.5, shape=4, scale=8)
## Not run: est <- dheaping(Sim3$xheaped, rounds=Sim3$rounds, boundary=TRUE, unequal=TRUE, setBias=T)
## End(Not run)

---

dshape3dProp

**3d Kernel density estimation for data classified in polygons or shapes**

**Description**

3d Kernel density estimation for data classified in polygons or shapes

**Usage**

dshape3dProp(
data,
burnin = 2,
samples = 5,
shapefile,
gridsize = 200,
boundary = FALSE,
deleteShapes = NULL,
fastWeights = TRUE,
numChains = 1,
numThreads = 1
)

**Arguments**

data
data.frame with 5 columns: x-coordinate, y-coordinate (i.e. center of polygon) and number of observations in area for partial population and number of observations for complete observations and third variable (numeric).
burnin
burn-in sample size
samples
sampling iteration size
shapefile
shapefile with number of polygons equal to nrow(data) / length(unique(data[,5]))
gridsize
number of evaluation grid points
boundary
boundary corrected kernel density estimate?
deleteShapes
shapefile containing areas without observations
fastWeights
if TRUE weights for boundary estimation are only computed for first 10 percent of samples to speed up computation
numChains
number of chains of SEM algorithm
numThreads
number of threads to be used (only applicable if more than one chains)
**dshapebivr**  

_Bivariate Kernel density estimation for data classified in polygons or shapes_

**Description**

Bivariate Kernel density estimation for data classified in polygons or shapes

**Usage**

```r
dshapebivr(  
data,  
burnin = 2,  
samples = 5,  
adaptive = FALSE,  
shapefile,  
gridsize = 200,  
boundary = FALSE,  
deleteShapes = NULL,  
fastWeights = TRUE,  
numChains = 1,  
numThreads = 1  
)
```

**Arguments**

- **data**
  - data.frame with 3 columns: x-coordinate, y-coordinate (i.e. center of polygon) and number of observations in area.
- **burnin**
  - burn-in sample size
- **samples**
  - sampling iteration size
- **adaptive**
  - TRUE for adaptive kernel density estimation
- **shapefile**
  - shapefile with number of polygons equal to nrow(data)
- **gridsize**
  - number of evaluation grid points
- **boundary**
  - boundary corrected kernel density estimate?
- **deleteShapes**
  - shapefile containing areas without observations
- **fastWeights**
  - if TRUE weights for boundary estimation are only computed for first 10 percent of samples to speed up computation
- **numChains**
  - number of chains of SEM algorithm
- **numThreads**
  - number of threads to be used (only applicable if more than one chains)
Value

The function returns a list object with the following objects (besides all input objects):

- `Mestimates`: kde object containing the corrected density estimate
- `gridx`: Vector Grid of x-coordinates on which density is evaluated
- `gridy`: Vector Grid of y-coordinates on which density is evaluated
- `resultDensity`: Matrix with Estimated Density for each iteration
- `resultX`: Matrix of true latent values X estimates

Examples

```r
## Not run:
library(maptools)

# Read Shapefile of Berlin Urban Planning Areas (download available from:
Berlin <- rgdal::readOGR("X:/SomeDir/RBS_OD_LOR_2015_12.shp") #(von daten.berlin.de)

# Get Dataset of Berlin Population (download available from:
# https://www.statistik-berlin-brandenburg.de/opendata/EWR201512E_Matrix.csv)
data <- read.csv2("X:/SomeDir/EWR201512E_Matrix.csv")

# Form Dataset for Estimation Process
dataIn <- cbind(t(sapply(1:length(Berlin@polygons),
  function(x) Berlin@polygons[[x]]@labpt)), data$E_E65U80)

# Estimate Bivariate Density
Est <- dshapebivr(data = dataIn, burnin = 5, samples = 10, adaptive = FALSE,
  shapefile = Berlin, gridsize = 325, boundary = TRUE)

## End(Not run)

# Plot Density over Area:
## Not run: breaks <- seq(1E-16,max(Est$Mestimates$estimate),length.out = 20)
image.plot(x=Est$Mestimates$eval.points[[1]],y=Est$Mestimates$eval.points[[2]],
  z=Est$Mestimates$estimate, asp=1, breaks = breaks,
  col = colorRampPalette(brewer.pal(9,"YlOrRd"))(length(breaks)-1))
plot(Berlin, add=TRUE)
## End(Not run)
```

dshapebivrProp

Bivariate Kernel density estimation for data classified in polygons or shapes

Description

Bivariate Kernel density estimation for data classified in polygons or shapes
Usage

dshapebivrProp(
  data,
  burnin = 2,
  samples = 5,
  adaptive = FALSE,
  shapefile,
  gridsize = 200,
  boundary = FALSE,
  deleteShapes = NULL,
  fastWeights = TRUE,
  numChains = 1,
  numThreads = 1
)

Arguments

data                   data.frame with 4 columns: x-coordinate, y-coordinate (i.e. center of polygon) and number of observations in area for partial population and number of observations for complete observations.
burnin                 burn-in sample size
samples                sampling iteration size
adaptive               TRUE for adaptive kernel density estimation
shapefile              shapefile with number of polygons equal to nrow(data)
gridsize               number of evaluation grid points
boundary                boundary corrected kernel density estimate?
deleteShapes           shapefile containing areas without observations
fastWeights            if TRUE weights for boundary estimation are only computed for first 10 percent of samples to speed up computation
numChains              number of chains of SEM algorithm
numThreads             number of threads to be used (only applicable if more than one chains)

Examples

## Not run:
library(maptools)

Berlin <- rgdal::readOGR("X:/SomeDir/RBS_OD_LOR_2015_12.shp") #(von daten.berlin.de)

data <- read.csv2("X:/SomeDir/EWR201512E_Matrix.csv")

# Form Dataset for Estimation Process
dataIn <- cbind(t(sapply(1:length(Berlin@polygons),
function(x) Berlin@polygons[[x]]@labpt)), data$E_E6SU80, data$E_E)

# Estimate Bivariate Proportions (may take some minutes)
PropEst <- dshapebivrProp(data = dataIn, burnin = 5, samples = 20, adaptive = FALSE,
shapefile = Berlin, gridsize=325, numChains = 16, numThreads = 4)
## End(Not run)

# Plot Proportions over Area:
## Not run:
breaks <- seq(0,0.4,by=0.025)
image.plot(x=PropEst$Mestimates$eval.points[[1]],y=PropEst$Mestimates$eval.points[[2]],
z=PropEst$proportion+1E-96, asp=1, breaks = breaks,
col = colorRampPalette(brewer.pal(9,"YlOrRd"))(length(breaks)-1))
plot(Berlin, add=TRUE)
## End(Not run)

---

**Kernelheaping**

*Kernel Density Estimation for Heaped Data*

**Description**

In self-reported or anonymized data the user often encounters heaped data, i.e. data which are rounded (to a possibly different degree of coarseness). While this is mostly a minor problem in parametric density estimation the bias can be very large for non-parametric methods such as kernel density estimation. This package implements a partly Bayesian algorithm treating the true unknown values as additional parameters and estimates the rounding parameters to give a corrected kernel density estimate. It supports various standard bandwidth selection methods. Varying rounding probabilities (depending on the true value) and asymmetric rounding is estimable as well. Additionally, bivariate non-parametric density estimation for rounded data is supported.

**Details**

The most important function is `dheaping`. See the help and the attached examples on how to use the package.

---

**plot.bivrounding**

*Plot Kernel density estimate of heaped data naively and corrected by partly bayesian model*

**Description**

Plot Kernel density estimate of heaped data naively and corrected by partly bayesian model

**Usage**

```r
## S3 method for class 'bivrounding'
plot(x, trueX = NULL, ...)
```
Arguments

x  bivrounding object produced by dbivr function
trueX optional, if true values X are known (in simulations, for example) the 'Oracle' density estimate is added as well
... additional arguments given to standard plot function

Value

plot with Kernel density estimates (Naive, Corrected and True (if provided))

plot.Kernelheaping  

Plot Kernel density estimate of heaped data naively and corrected by partly bayesian model

Description

Plot Kernel density estimate of heaped data naively and corrected by partly bayesian model

Usage

## S3 method for class 'Kernelheaping'
plot(x, trueX = NULL, ...)

Arguments

x  Kernelheaping object produced by dheaping function
trueX optional, if true values X are known (in simulations, for example) the 'Oracle' density estimate is added as well
... additional arguments given to standard plot function

Value

plot with Kernel density estimates (Naive, Corrected and True (if provided))
**Description**

Simulation of heaping correction method

**Usage**

```r
sim.Kernelheaping(
    simRuns,
    n,
    distribution,
    rounds,
    thresholds,
    downbias = 0.5,
    setBias = FALSE,
    Beta = 0,
    unequal = FALSE,
    burnin = 5,
    samples = 10,
    bw = "nrd0",
    offset = 0,
    boundary = FALSE,
    adjust = 1,
    ...
)
```

**Arguments**

- `simRuns`: number of simulations runs
- `n`: sample size
- `distribution`: name of the distribution where random sampling is available, e.g. "norm"
- `rounds`: rounding values, numeric vector of length >=1
- `thresholds`: rounding thresholds
- `downbias`: Bias parameter used in the simulation
- `setBias`: if TRUE a rounding Bias parameter is estimated. For values above 0.5, the respondents are more prone to round down, while for values < 0.5 they are more likely to round up
- `Beta`: Parameter of the probit model for rounding probabilities used in simulation
- `unequal`: if TRUE a probit model is fitted for the rounding probabilities with \log(\text{true value}) as regressor
- `burnin`: burn-in sample size
- `samples`: sampling iteration size
bw | bandwidth selector method, defaults to "nrd0" see density for more options
offset | location shift parameter used simulation in simulation
boundary | TRUE for positive only data (no positive density for negative values)
adjust | as in density, the user can multiply the bandwidth by a certain factor such that \( bw = \text{adjust} \times bw \)
... | additional attributes handed over to createSim.Kernelheaping

Value

List of estimation results

Examples

```r
## Not run: Sims1 <- sim.Kernelheaping(simRuns=2, n=500, distribution="norm",
rounds=c(1,10,100), thresholds=c(0.3,0.4,0.3), sd=100)
## End(Not run)
```

---

## Description

Simulation Summary

## Usage

```r
simSummary.Kernelheaping(sim, coverage = 0.9)
```

## Arguments

- `sim` Simulation object returned from sim.Kernelheaping
- `coverage` probability for computing coverage intervals

## Value

list with summary statistics
Data collected during 2004 and 2005 from students in statistics classes at a large state university in the northeastern United States.

Source

http://mathfaculty.fullerton.edu/mori/Math120/Data/readme

References


Summary

Prints some descriptive statistics (means and quantiles) for the estimated rounding, bias and acceleration (beta) parameters

Usage

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

Arguments

- `object`: Kernelheaping object produced by dheaping function
- `...`: unused

Value

Prints summary statistics
toOtherShape

Description
Transfer observations to other shape

Usage
toOtherShape(Mestimates, shapefile)

Arguments
Mestimates      Estimation object created by functions dshapebivr and dbivr
shapefile      The new shapefile for which the observations shall be transferred to

Value
The function returns the count, sd and 90

tracePlots
Plots some trace plots for the rounding, bias and acceleration (beta) parameters

Description
Plots some trace plots for the rounding, bias and acceleration (beta) parameters

Usage
tracePlots(x, ...)

Arguments
x      Kernelheaping object produced by dheaping function
...    additional arguments given to standard plot function

Value
Prints summary statistics
Index

createSim.Kernelheaping, 2

dbivr, 3
dclass, 4
dheaping, 6, 12
dshape3dProp, 8
dshapebivr, 9
dshapebivrProp, 10

Kernelheaping, 12

plot.bivrounding, 12
plot.Kernelheaping, 13

sim.Kernelheaping, 14
simSummary.Kernelheaping, 15
students, 16
summary.Kernelheaping, 16

toOtherShape, 17
tracePlots, 17