Package ‘GFE’

August 2, 2018

Type Package

Title Gross Flows Estimation under Complex Surveys

Version 0.1.0

Author Acero William <wfaceror@unal.edu.co>, Gutiérrez Andrés <andres.gutierrez@un.org>, Trujillo Leonardo <ltrujillo@unal.edu.co>

Maintainer Acero William <wfaceror@unal.edu.co>


Depends R (>= 3.3.1)

License GPL (>= 2)

Encoding latin1

LazyData true

RoxygenNote 6.0.1

Repository CRAN

Imports dplyr, data.table, TeachingSampling

NeedsCompilation no

Date/Publication 2018-08-02 12:10:10 UTC

R topics documented:

createBase ......................................................... 2
estGF ............................................................. 3
reSamGF ............................................................ 5
round_preserve_sum ............................................. 8

Index 9
createBase  
*Create a database for gross flows.*

**Description**
Create a database based on $\xi$ model.

**Usage**
createBase(x)

**Arguments**

$x$  
A matrix that contains information of the observable process.

**Value**
createBase returns `data.frame` that contains the database based on $\xi$ model.

**Examples**

candidates_t0 <- c("Candidate1", "Candidate2", "Candidate3", "Candidate4", 
"Candidate5", "WhiteVote", "NoVote")
candidates_t1 <- c("Candidate3", "Candidate5", "WhiteVote", "NoVote")

N  <- 100000
nCanT0  <- length(candidates_t0)
nCanT1  <- length(candidates_t1)

etta  <- matrix(c(0.10, 0.10, 0.20, 0.17, 0.28, 0.1, 0.05),
byrow = TRUE, nrow = nCanT0)
P  <- matrix(c(0.10, 0.60, 0.15, 0.15,
0.30, 0.10, 0.25, 0.35,
0.34, 0.25, 0.16, 0.25,
0.25, 0.05, 0.35, 0.35,
0.10, 0.25, 0.45, 0.20,
0.12, 0.36, 0.22, 0.30,
0.10, 0.15, 0.30, 0.45),
byrow = TRUE, nrow = nCanT0)
citaModel  <- matrix(), ncol = nCanT1, nrow = nCanT0)
row.names(citaModel) <- candidates_t0
colnames(citaModel) <- candidates_t1

for(ii in 1:nCanT0){
  citaModel[ii,] <- c(rmultinom(1, size = N * etta[ii,], prob = P[ii,]))}

# # Model I
psiI  <- 0.9
estGF <- Gross Flows estimation

Description

Gross Flows under complex electoral surveys.

Usage

estGF(sampleBase = NULL, niter = 100, model = NULL, colWeights = NULL, nonrft = FALSE)

Arguments

sampleBase An object of class "data.frame" containing the information of electoral candidates. The data must contain the samplings weights.

niter The number of iterations for the \( \eta_i \) and \( p_{ij} \) model parameters within the model.

model A character indicating the model to be used in estimating estimated gross flows. The models available are: "I","II","III","IV" (see also "Details").

colWeights The column name containing the sampling weights to be used in the fitting process.

nonrft A logical value indicating a non response for first time.

Details

The population size \( N \) must satisfy the condition:

\[
N = \sum_j \sum_i N_{ij} + \sum_j C_j + \sum_i R_i + M
\]

where, \( N_{ij} \) is the amount of people interviewed who have classification \( i \) at first time and classification \( j \) at second time, \( R_i \) is the amount of people who did not respond at second time, but did at first time, \( C_j \) is the amount of people who did not respond at first time, but they did at second time
and \( M \) is the number of people who did not respond at any time or could not be reached. Let \( \eta_i \) the initial probability that a person has classification \( i \) in the first time, and let \( p_{ij} \) the vote transition probability for the cell \((i,j)\), where \( \sum_i \eta_i = 1 \) and \( \sum_j p_{ij} = 1 \). Thus, four possibles models for the gross flows are given by:

1. **Model I**: This model assumes that a person’s initial probability of being classified as \( i \) at first time is the same for everyone, that is, \( \psi(i,j) = \psi \). Besides, transition probabilities between respond and non response not depend of the classification \((i,j)\), that is \( \rho_{MM}(i,j) = \rho_{MM} \) and \( \rho_{RR}(i,j) = \rho_{RR} \).

2. **Model II**: Unlike ‘Model I’, this model assumes that person initial probability that person has classification \((i,j)\), only depends of his classification at first time, that is \( \psi(i,j) = \psi(i) \).

3. **Model III**: Unlike ‘Model I’, this model assumes that transition probabilities between response and non response only depends of probability classification at first time, that is \( \rho_{MM}(i,j) = \rho_{MM}(i) \) and \( \rho_{RR}(i,j) = \rho_{RR}(i) \).

4. **Model IV**: Unlike ‘Model I’, this model assumes that transition probabilities between response and non response only depends of probability classification at second time, that is \( \rho_{MM}(i,j) = \rho_{MM}(j) \) and \( \rho_{RR}(i,j) = \rho_{RR}(j) \).

**Value**

`estGF` returns a list containing:

1. **Est.CIV**: a data.frame containing the gross flows estimation.
2. **Params.Model**: a list that contains the \( \hat{\eta}_i \), \( \hat{p}_{ij} \), \( \hat{\psi}(i,j) \), \( \hat{\rho}_{RR}(i,j) \), \( \hat{\rho}_{MM}(i,j) \) parameters for the estimated model.
3. **Sam.Est**: a list containing the sampling estimators \( \hat{N}_{ij} \), \( \hat{R}_i \), \( \hat{C}_j \), \( \hat{M} \), \( \hat{N} \).

**References**


**Examples**

```r
library(TeachingSampling)
library(data.table)
# Colombia's electoral candidates in 2014
candidates_t0 <- c("Clara","Enrique","Santos","Martha","Zuluaga","WhiteVote","NoVote")
candidates_t1 <- c("Santos","Zuluaga","WhiteVote","NoVote")

N <- 100000
mCanT0 <- length(candidates_t0)
mCanT1 <- length(candidates_t1)
# Initial probabilities
eta <- matrix(c(0.10, 0.10, 0.20, 0.17, 0.28, 0.1, 0.05),
```
byrow = TRUE, nrow = nCanT0)
  # Transition probabilities
  P <- matrix(c(0.10, 0.60, 0.15, 0.15,
                0.30, 0.10, 0.25, 0.35,
                0.34, 0.25, 0.16, 0.25,
                0.25, 0.35, 0.35, 0.20,
                0.10, 0.25, 0.45, 0.20,
                0.12, 0.36, 0.22, 0.30,
                0.10, 0.15, 0.30, 0.45),
            byrow = TRUE, nrow = nCanT0)
citaMod <- matrix(nrow = nCanT1, nrow = nCanT0)
row.names(citaMod) <- candidates_t0
colnames(citaMod) <- candidates_t1

for(ii in 1:nCanT0){
citaMod[ii,] <- c(rmultinom(1, size = N * eta[ii,], prob = P[ii,]))
}

# Model I
psiI <- 0.9
rhoRRI <- 0.9
rhoMII <- 0.5

citaModI <- matrix(nrow = nCanT0 + 1, ncol = nCanT1 + 1)
rownames(citaModI) <- c(candidates_t0, "Non RESP")
colnames(citaModI) <- c(candidates_t1, "Non RESP")
citaModI[1:nCanT0, 1:nCanT1] <- P * c(eta) * rhoRRI * psiI
citaModI[(nCanT0 + 1), (nCanT1 + 1)] <- rhoMII * (1 - psiI)
citaModI[1:nCanT0, (nCanT1 + 1)] <- (1 - rhoRRI) * psiI * rowSums(P * c(eta))
citaModI[(nCanT0 + 1), 1:nCanT1] <- (1 - rhoMII) * (1 - psiI) * colSums(P * c(eta))
citaModI <- round_preserve_sum(citaModI * N)

# Creating auxiliary information
DBcitaModI[, AuxVar := rnorm(nrow(DBcitaModI), mean = 45, sd = 10)]

# Selects a sample with unequal probabilities
res <- s.psiP(n = 3200, as.data.frame(DBcitaModI)[, "AuxVar"])
sam <- res[, 1]
pik <- res[, 2]
DBcitaModISam <- copy(DBcitaModI[sam,])

# Gross flows estimation
estima <- estGF(sampleBase = DBcitaModISam, niter = 500, model = "I", colWeights = "Pik")
estima

gross flows variance estimation.
Description

Gross flows variance estimation according to resampling method (Bootstrap or Jackknife).

Usage

reSamGF(sampleBase = NULL, nRepBoot = 500, model = "I", niter = 100,
         type = "Bootstrap", colWeights = NULL, nonrft = FALSE)

Arguments

- `sampleBase`: An object of class data.frame or data.table containing the sample selected to estimate the gross flows.
- `nRepBoot`: The number of replicates for the bootstrap method.
- `model`: A character indicating the model that will be used for estimate the gross flows. The available models are: 'I','II','III','IV'.
- `niter`: The number of iterations for the $\eta_i$ and $p_{ij}$ model parameters.
- `type`: A character indicating the resampling method ("Bootstrap" or "Jackknife")
- `colWeights`: The data column name containing the sampling weights to be used on the fitting process.
- `nonrft`: a logical value indicating the non response for the first time.

Details

The resampling methods for variance estimation are:

- **Bootstrap**: This technique allows to estimate the sampling distribution of almost any statistic by using random sampling methods. Bootstrapping is the practice of estimating properties of an statistic (such as its variance) by measuring those properties from it's approximated sample.

- **Jackknife**: The jackknife estimate of a parameter is found by systematically leaving out each observation from a dataset and calculating the estimate and then finding the average of these calculations. Given a sample of size $n$, the jackknife estimate is found by aggregating the estimates of each $n-1$-sized sub-sample.

Value

`reSamGF` returns a list that contains the variance of each parameter of the selected model.

References

Examples

```r
library(TeachingSampling)
library(data.table)

# Colombia's electoral candidates in 2014
candidates_t0 <- c("Clara", "Enrique", "Santos", "Martha", "Zuluaga", "Blanco", "NoVoto")
candidates_t1 <- c("Santos", "Zuluaga", "Blanco", "NoVoto")

N <- 100000
nCanT0 <- length(candidates_t0)
nCanT1 <- length(candidates_t1)

# Initial probabilities
eta <- matrix(c(0.10, 0.10, 0.20, 0.17, 0.28, 0.1, 0.05), byrow = TRUE, nrow = nCanT0)
# Transition probabilities
P <- matrix(c(0.10, 0.60, 0.15, 0.15,
              0.30, 0.10, 0.25, 0.35,
              0.34, 0.25, 0.16, 0.25,
              0.25, 0.05, 0.35, 0.35,
              0.10, 0.25, 0.45, 0.20,
              0.12, 0.36, 0.22, 0.30,
              0.10, 0.15, 0.30, 0.45), byrow = TRUE, nrow = nCanT0)

citaMod <- matrix(, ncol = nCanT1, nrow = nCanT0)
row.names(citaMod) <- candidates_t0
colnames(citaMod) <- candidates_t1

for(ii in 1:nCanT0){
  citaMod[ii,] <- c(rmultinom(1, size = N * eta[ii,], prob = P[ii,]))
}

# # Model I
psiI <- 0.9
rhorrI <- 0.9
rhommi <- 0.5

citaModI <- matrix(nrow = nCanT0 + 1, ncol = nCanT1 + 1)
rownames(citaModI) <- c(candidates_t0, "Non_Resp")
colnames(citaModI) <- c(candidates_t1, "Non_Resp")

citaModI[1:nCanT0, 1:nCanT1] <- P * c(eta) * rhorrI * psiI
citaModI[nCanT0 + 1, (nCanT1 + 1)] <- rhommi * (1-psiI)
citaModI[1:nCanT0, (nCanT1 + 1)] <- (1-rhorrI) * psiI * rowSums(P * c(eta))
citaModI[(nCanT0 + 1), 1:nCanT1] <- (1-rhommi) * (1-psiI) * colSums(P * c(eta))
citaModI <- round_preserve_sum(citaModI * N)

DBcitaModI <- createBase(citaModI)

# Creating auxiliary information
DBcitaModI[, AuxVar := rnorm(nrow(DBcitaModI), mean = 45, sd = 10)]
# Selects a sample with unequal probabilities
res <- S.pIPS(n = 1200, as.data.frame(DBcitaModI)[, "AuxVar"])
```

round_preserve_sum

Description

Rounds a vector of numbers while preserving the sum of them.

Usage

round_preserve_sum(x, digits = 0)

Arguments

x A numeric vector.
digits The number of digits to take in account in the rounding process.

Value

round_preserve_sum returns y with round vector.

Source

https://www.r-bloggers.com/round-values-while-preserve-their-rounded-sum-in-r/ and

Examples

sum(c(0.333, 0.333, 0.334))
round(c(0.333, 0.333, 0.334), 2)
sum(round(c(0.333, 0.333, 0.334), 2))
round_preserve_sum(c(0.333, 0.333, 0.334), 2)
sum(round_preserve_sum(c(0.333, 0.333, 0.334), 2))
Index

createBase, 2
estGF, 3
reSamGF, 5
round_preserve_sum, 8