Package ‘GRCdata’

August 20, 2017

Type Package

Title Parameter Inference and Optimal Designs for Grouped and/or Right-Censored Count Data

Version 1.0

Date 2017-07-28

Author Xin Guo <x.guo@polyu.edu.hk>, Qiang Fu <qiang.fu@ubc.ca>

Maintainer Xin Guo <x.guo@polyu.edu.hk>

Depends nloptr, cubature

Description We implement two main functions.
The first function uses a given grouped and/or
right-censored grouping scheme and empirical data to infer parameters,
and implements chi-square goodness-of-fit tests.
The second function searches for the global optimal grouping
scheme of grouped and/or right-censored count responses in surveys.

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-20 16:47:38 UTC

R topics documented:

GRCdata-package	*Parameter inference and optimal designs for grouped and/or right-censored count data*

Description

This package consists of two main functions: The first function uses a given grouped and/or right-censored grouping scheme and empirical data to infer parameters, and implements chi-square goodness-of-fit tests; The second function searches for the global optimal grouping scheme of grouped and/or right-censored count responses in surveys.

This R package is designed to implement methods and algorithms developed in the following papers and please cite these articles at your convenience:

Qiang Fu, Xin Guo and Kenneth C. Land. Forthcoming. "A Poisson-Multinomial Mixture Approach to Grouped and Right-Censored Counts." Communications in Statistics – Theory and Methods. DOI: 10.1080/03610926.2017.1303736 (mainly about the first function for aggregate-level parameter inference)

Qiang Fu, Xin Guo and Kenneth C. Land. Conditionally accepted. "Optimizing Count Responses in Surveys: A Machine-Learning Approach." Sociological Methods & Research. (mainly about the second function for finding optimal grouping schemes)

To install the package "GRCdata_1.0.tar.gz", one may place this file in the working directory/folder of R, and type

```
install.packages("GRCdata", repos = NULL, type = "source")
```

To check the current working directory of R, one may type

```
getwd()
```

To see the source code, one could extract the package “GRCdata_1.0.tar.gz”. There would be two directories/folders: `man` and `R`. The source code is under the `R` directory/folder.

Details

Package:	GRCdata
Type:	Package
Version:	1.0
Date:	July 28, 2017
License:	GPLv3

Author(s)

Authors: Xin Guo <x.guo@polyu.edu.hk>, Qiang Fu <qiang.fu@ubc.ca>

Maintainers: Xin Guo <x.guo@polyu.edu.hk>

References

Qiang Fu, Xin Guo and Kenneth C. Land. Conditionally accepted. "Optimizing Count Responses in Surveys: A Machine-Learning Approach." Sociological Methods & Research.

Qiang Fu, Xin Guo and Kenneth C. Land. Forthcoming. "A Poisson-Multinomial Mixture Approach to Grouped and Right-Censored Counts." Communications in Statistics – Theory and Methods. DOI: 10.1080/03610926.2017.1303736

`find.scheme` *Search for the global optimal grouping scheme of grouped and/or right-censored count data*

Description

Given the prior distribution (or values) of parameters, and the total/maximum number of groups (N) allowed for grouping schemes, this function finds the global optimal grouping scheme that makes the sampling process most informative.

Usage

```
find.scheme(N,
  densityFUN, lambda.lwr, lambda.upr, p.lwr, p.upr,
  probs, lambdas, ps,
  is.0.isolated = TRUE, model = c("Poisson", "ZIP"),
  matSc = c("A", "D", "E"), M = "auto")
```

Arguments

`N` (maximum) number of groups allowed for all grouping schemes. A non-integral value will be coerced to an integer.

`densityFUN, lambda.lwr, lambda.upr, p.lwr, p.upr`
prior information of parameters in a continuous form. These parameters denote the prior probability density function (optional), the lower bound of λ (for Poisson models), the higher bound of λ (for Poisson models), the lower bound of `p` (optional for ZIP models), the higher bound of `p` (optional for ZIP models), respectively.

`probs, lambdas, ps`
prior information of the parameters in a discrete form. These parameters are vectors denoting the mass probabilities, the corresponding values of λ and `p` (optional), respectively.

`is.0.isolated` a logical value indicating whether zero is contained and only contained in a single group.

`model` underlying Poisson models to be used for optimal designs: `Poisson` or `ZIP`. The default value is `Poisson`.

matSc
: A character indicating types of optimality functions of the Fisher information (matrix). It must be one from the three letters: A, D, and E. In particular, if J is the 2-by-2 Fisher information matrix, then

"A" or A-optimality maximizes $1/\mathrm{tr}(J^{-1})$;

"D" or D-optimality maximizes $\det(J)$;

"E" or E-optimality maximizes the minimum eigenvalue of J.

M
: a sufficiently large integer needed to facilitate the search, or a character "auto". Theoretically, it could be the lowest integer contained in the last right-censored group of the global optimal grouping scheme. A non-integral value will be coerced to an integer. If M is set to be "auto", the algorithm takes longer time to converge because it will automatically determine M and return the global optimal grouping scheme; The default value of M is "auto".

Details

This function tries to find the N-group scheme maximizing Fisher information (matrix). If model is specified as Poisson, p.lwr or p.upr will be ignored. When the prior distribution is discrete, lambdas specify discrete values that λ may take, and probs specify probabilities associated with p. In the ZIP model, lambdas and ps specify discrete values that λ and p may take, respectively. probs denotes joint mass probabilities associated with (λ, p). The values of (p.lwr, p.upr) cannot be (0, 1) as the algorithm will not converge. Instead, approximate values, such as (0.000001, 0.999999), can be used.

A sufficiently large integer M should be provided by the user so that infinitely many grouping schemes could be handled by the search algorithm. M is in theory the lowest integer to be contained in the last right-censored group of the global optimal grouping scheme. In practice, the choice of M should be slightly higher than its theoretical value because the search algorithem is designed in a way that it prevents any acceptance of a false optimal solution at the cost of tolerating false rejection of the correct optimal grouping scheme. This idea is implemented by a logical indicator succeed in the output. Its value will be TRUE if the real optimal grouping scheme is identified. Otherwise, a FALSE output means that M is not large enough to gurantee that the grouping scheme yielded by the search algorithm is the global optimal grouping scheme. Researchers then need to select a larger M and repeat this process until the logical indicator succeed becomes TRUE. Alternatively, users may use the "auto" option so that this iterative process will be automatically implemented.

Value

The returned value is a list with components.

best.scheme.compact, best.scheme.loose, best.scheme.innerCode
: the same optimal grouping scheme is printed in various forms.

succeed
: see Details. This is a logical variable. The global optimal grouping scheme is obtained if it is TRUE; a larger M needs to be selected for a successful search if it is FALSE.

Author(s)

Xin Guo <x.guo@polyu.edu.hk>, Qiang Fu <qiang.fu@ubc.ca>

References

Qiang Fu, Xin Guo and Kenneth C. Land. Conditionally accepted. "Optimizing Count Responses in Surveys: A Machine-Learning Approach." Sociological Methods & Research.

Examples

```
# Example 1 ####################################
# M=7, N=3, 0 is not required to be contained
# in a separate group of grouping schemes.
# Poisson model, lambda takes 4 and 5 and each value has a probability of 0.5.
find.scheme(probs = c(0.5, 0.5), lambdas = c(4,5),
  M = 7, N = 3, is.0.isolated = FALSE, model = "Poisson")

# Example 2 ####################################
# N=3, 0 is required to be contained in a separate group of grouping schemes.
# Poisson model, lambda takes 4 and 5 and each value has a probability of 0.5.
# M is not given, so it will be selected automatically.
find.scheme(probs = c(0.5, 0.5), lambdas = c(4,5),
  N = 3, is.0.isolated = TRUE, model = "Poisson")

# Example 3 ####################################
# M=7, N=3, 0 is not required to be contained in a separate group.
# ZIP model, (lambda, p) take (4, 0.3) and (5, 0.4)
# with their probabilities denoted by c(0.5, 0.5)

find.scheme(probs = c(0.5, 0.5), lambdas = c(4,5), ps = c(0.3, 0.5),
  M = 7, N = 3, is.0.isolated = FALSE, model = "ZIP")


# Example 4 ####################################
# N=3, 0 is not required to be contained in a separate group.
# Poisson model, lambda takes a normal distribution truncated to [1, 10]
# M is not given, so it will be selected automatically.

find.scheme(densityFUN = function(lambda)
  dnorm(lambda, mean = 3, sd = 1),
  lambda.lwr = 1, lambda.upr = 10,
  N = 3, is.0.isolated = FALSE, model = "Poisson")


# Example 5 ####################################
# M=7, N=3, 0 is required to be contained in a separate group.
# Poisson model, lambda takes a normal distribution truncated to [1, 10]

find.scheme(densityFUN = function(lambda)
  dnorm(lambda, mean = 3, sd = 1),
  lambda.lwr = 1, lambda.upr = 10,
  M = 7, N = 3, is.0.isolated = TRUE, model = "Poisson")


# Example 6 ####################################
# N=3, 0 is required to be contained in a separate group.
```

```
# Poisson model, lambda takes an uniform distribution on [1, 10]
# M is not given, so it will be selected automatically.
find.scheme(densityFUN = function(lambda)
  dunif(lambda, min = 1, max = 10),
  lambda.lwr = 1, lambda.upr = 10,
  N = 3, is.0.isolated = TRUE, model = "Poisson")

# Example 7 #################################
# M=7, N=3, 0 is required to be contained in a separate group.
# ZIP model, (lambda, p) has an uniform distribution with
# lambda on [1,10] and p on [0.1, 0.9]

find.scheme(densityFUN = function(...) 1,
  lambda.lwr = 1, lambda.upr = 10, p.lwr = 0.0001, p.upr = 0.9999,
  M = 7, N = 3, is.0.isolated = TRUE, model = "ZIP")


# Example 8 ####################################
# M=7, N=3, 0 is required to be contained in a separate group.
# ZIP model, (lambda, p) has a normal distribution centered
# at (5.5, 0.5) with a covariance matrix showing their correlation
#    /                 \
#    |  11/3      3    |
#    |    3     11/3   |
#    \                 /.
# This normal distribution is also truncated to
# [1, 10] X [0.1, 0.9]
# Note: this example may take several minutes to converge,
# depending on your computer configuration.

dsty <- function(lambda, p){
  vec <- c(lambda - 5.5, p - 0.5)
  mat <- matrix(c(11/3,3,3,11/3), nrow = 2, ncol = 2)
  pw <- -0.5 * sum(vec * solve(mat, vec))
  return(exp(pw))
}
find.scheme(densityFUN = dsty,
  lambda.lwr = 1, lambda.upr = 10, p.lwr = 0.1, p.upr = 0.9,
  M = 7, N = 3, is.0.isolated = TRUE, model = "ZIP")
```

`grcmle` *Maximum likelihood estimation of Poisson or ZIP parameters at the aggregate level.*

Description

This function infers Poisson or zero-inflated Poisson (ZIP) parameters from grouped and right-censored count data, and conducts a chi-squared goodness-of-fit test. A grouped and right-censored scheme may look like

`0, 1, 2--4, 5--8, 9+.`

For grouped and right-censored count data collected in a survey, such as frequency of alcohol drinking, number of births or occurrence of crimes, the response category designed as the example above means never, once, 2 to 4 times, 5 to 8 times, 9 times and more. The frequency distribution from a sample corresponding to the example above may look like

`3, 15, 168, 155, 15.`

Usage

```
grcmle(counts, scheme, method = c("Poisson", "ZIP"),
  do.plot = T, init.guess = NULL,
  optimizing.algorithm.index = 2, lambda.extend.ratio = 3,
  conf.level = 0.95)
```

Arguments

`counts` specifies the frequency distribution of the grouped and right-censored count data. For the example above, one may input

`counts = c(3, 15, 168, 155, 15).`

`scheme` specifies the grouping scheme. It should be a vector of integers containing the starting point (or the lowest integer) of each group. For example, to input the scheme above

`0, 1, 2--4, 5--8, 9+,`

one may use

`scheme = c(0, 1, 2, 5, 9).`

`method` a string parameter specifies which statistical model to use. Currently there are two options `"Poisson"` and `"ZIP"`. The default value is `"Poisson"`. It can be abbreviated.

`do.plot` a logical variable indicating whether or not to plot the log likelihood. The default is `T`.

`init.guess` the initial value used for the optimization procedure of the likelihood estimation. The default value is `NULL`, which instructs the function `grcmle` to select the initial value automatically.

`optimizing.algorithm.index`

defines which optimization algorithm to use. Currently the possible values are `1,2,3,4,5,6,7` and `8`, representing the following algorithms, respectively:

`NLOPT_GN_DIRECT_L`

`NLOPT_GN_DIRECT`

`NLOPT_GN_DIRECT_L_RAND`

`NLOPT_GN_DIRECT_NOSCAL`

`NLOPT_GN_DIRECT_L_NOSCAL`

`NLOPT_GN_DIRECT_L_RAND_NOSCAL`

`NLOPT_GN_ORIG_DIRECT`

`NLOPT_GN_ORIG_DIRECT_L`

For details of these algorithms, please see the manual of the R package `"nloptr"`. The default value is 2.

lambda.extend.ratio
: specifies the searching interval of possible λ as $[0, nr]$, where n is the left end (i.e., the lowest integer) of the last right-censored group, and r is `lambda.extend.ratio`. By default, we set

  `lambda.extend.ratio=3`.

conf.level
: confidence level of the confidence interval(s) for the parameter(s) inferred

Details

Maximum likelihood estimation is used for the inference.

Value

The returned value is a list containing

mle
: the parameter(s) inferred. For Poisson model, it is the estimate of λ. For ZIP model, it shows a vector of length 2: the first element is the estimate of p and the second element is the estimate of λ.

p.value
: the p-value of the chi-squared test of goodness-of-fit.

df
: the degree(s) of freedom of the chi-squared test of goodness-of-fit.

CI.lambda
: the confidence interval of λ obtained by normal approximation

CI.p
: the confidence interval of p obtained by normal approximation

conf.level
: the confidence level

std.err
: the standard error of λ or the standard errors of (p, λ), if a ZIP model is specified

Author(s)

Authors: Xin Guo <x.guo@polyu.edu.hk>, Qiang Fu <qiang.fu@ubc.ca>

Examples

```
grcmle(counts=c(6, 15, 168, 155, 15), scheme = c(0, 1, 2, 5, 9))

grcmle(counts=c(6, 15, 168, 155, 15), scheme = c(0, 1, 2, 5, 9), method = "ZIP")
```

Index