Package ‘csurvey’

October 12, 2022

Title Constrained Regression for Survey Data

Version 1.4


License GPL (>= 2)

Depends R (>= 4.0), survey (>= 3.36), cgam (>= 1.7), purrr(>= 0.3.4), MASS (>= 7.3-51.4)

Imports coneproj, graphics, grDevices, Matrix, tidyselect, dplyr, stats

ByteCompile true

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Author Xiyue Liao [aut, cre] (<https://orcid.org/0000-0002-4508-9219>)

Maintainer Xiyue Liao <xiyue.liao@csulb.edu>

Date/Publication 2022-08-28 02:30:02 UTC

R topics documented:

  block.Ord ................................................................. 2
  csvy ................................................................. 3
  nhdat ............................................................... 10

Index 11
block.Ord

Specify a Block Monotonic Shape-Restriction in a CSVY Formula

Description

A symbolic routine to define that a vector of domain means follows a monotonic ordering in a predictor in a formula argument to csvy. This is the unsmoothed version.

Usage

block.Ord(x, order = NULL, numknots = 0, knots = 0, space = "E")

Arguments

x
A numeric predictor which has the same length as the response vector.

order
A $1 \times M$ vector defining the order of domains when the shape constraint is block ordering.

numknots
The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.

knots
The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.

space
A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Value

The vector x with five attributes, i.e., name: the name of x; shape: 9("block ordering"); numknots: the numknots argument in "block.Ord"; knots: the knots argument in "block.Ord"; space: the space argument in "block.Ord".

Author(s)

Xiyue Liao

See Also

incr, decri
csvy

Design-based estimation of domain means with monotonicity constraints.

Description

The csvyby function performs design-based domain mean estimation with monotonicity and block-monotone shape constraints.

Usage

csvy(formula, design, subset = NULL, family = stats::gaussian(),
    nD = NULL, level = 0.95, n.mix = 100L, test = TRUE,...)

Arguments

- **formula**: A formula object which gives a symbolic description of the model to be fitted. It has the form "response ~ predictor". The response is a vector of length \( n \). A predictor can be a non-parametrically modelled variable with a monotonicity or block ordering restriction, or a combination of both. In terms of a non-parametrically modelled predictor, the user is supposed to indicate the relationship between the domain mean and a predictor \( x \) in the following way:
Assume that $\mu$ is the vector of domain means and $x$ is a predictor:

- incr($x$): $\mu$ is increasing in $x$.
- decr($x$): $\mu$ is decreasing in $x$.
- block.Ord($x$): $\mu$ has a block ordering in $x$.
- conc($x$): $\mu$ is concave in $x$.
- conv($x$): $\mu$ is convex in $x$.
- incr.conc($x$): $\mu$ is increasing and concave in $x$.
- incr.conv($x$): $\mu$ is increasing and convex in $x$.
- decr.conc($x$): $\mu$ is decreasing and concave in $x$.
- decr.conv($x$): $\mu$ is decreasing and convex in $x$.

**design**
A survey design, which must be specified by the svydesign routine in the survey package.

**subset**
Expression to select a subpopulation.

**nD**
Total number of domains.

**family**
A parameter indicating the error distribution and link function to be used in the model. It can be a character string naming a family function or the result of a call to a family function. This is borrowed from the glm routine in the stats package. There are four families: Gaussian, binomial, poisson, and Gamma.

**level**
Confidence level of the approximate confidence surfaces. The default is 0.95.

**n.mix**
The number of simulations used to get the approximate confidence intervals or surfaces. If $n.mix = 0$, no simulation will be done and the face of the final projection will be used to compute the covariance matrix of the constrained estimate. The default is $n.mix = 100L$.

**test**
A logical scalar. If test == TRUE, then the p-value for the test $H_0 : \theta$ is in $V$ versus $H_1 : \theta$ is in $C$ is returned. $C$ is the constraint cone of the form $\{\beta : A\beta \geq 0\}$, and $V$ is the null space of $A$. The default is test = TRUE.

This term includes two other arguments: deff and multicore. deff = TRUE will request a design effect from svymean. multicore = TRUE will use multicore package to distribute subsets over multiple processors.

The coef function returns estimated systematic component of a csvy object.

The confint function returns the confidence interval of a csvy object. If type = "response", then the interval is for the mean; if type = "link", then the interval is for the systematic component.

**parm**
An argument in the generic confint function in the stats package. For now, this argument is not in use.

The following arguments are used in the predict function.

**object**
A csvy object.

**newdata**
A data frame in which to look for variables with which to predict. If omitted, the fitted values are used.

**type**
If the response is Gaussian, type = "response" and type = "link" give the predicted mean; if the response is binomial, poisson or Gamma, type = "response" gives the predicted mean, and type = "link" gives the predicted systematic component.
Logical switch indicating if confidence intervals are required.
The following arguments are used in the \texttt{barplot} function. See \texttt{barplot.svystat} for more details.

Analysis result.

Grouped, rather than stacked, bars.
The following arguments are used in the \texttt{ftable} function. See \texttt{ftable.svystat} for more details.

A \texttt{csvy} object.
The following arguments are used in the \texttt{svycontrast} function. See \texttt{svycontrast} for more details.

A \texttt{csvy} object.

A vector or list of vectors of coefficients, or a call or list of calls.

Details

In a one dimensional situation, we assume that $\hat{y}_{S_t}$ are non-decreasing over $T$ domains. If this
monotonicity is not used in estimation, the population domain means can be estimated by the
Horvitz-Thompson estimator or the Hajek estimator. To use the monotonicity information, this
\texttt{csvy} function starts from the Hajek estimates $\hat{y}_{S_t} = \left(\sum_{k \in S_t} y_k / \pi_k\right) / N_t$ and the isotonic estimator
$(\hat{\theta}_1, \ldots, \hat{\theta}_T)^T$ minimizes the weighted sum of squared deviations from the sample domain means
over the set of ordered vectors; that is, $\hat{\theta}$ is the minimizer of $(\hat{y}_{S_t} - \theta)^T W_s (\hat{y}_{S_t} - \theta)$ subject to
$A\theta \geq 0$, where $W_s$ is the diagonal matrix with elements $\hat{N}_1 / N, \ldots, \hat{N}_D / N$, and $\hat{N} = \sum_{t=1}^T \hat{N}_t$
and $A$ is a $m \times T$ constraint matrix imposing the monotonicity constraint.

Domains can also be formed from multiple covariates. In that case, a grid will be used to represent
the domains. For example, if there are two predictors $x_1$ and $x_2$, and $x_1$ has values on $D_1$ domains:
1, \ldots, $D_1$, $x_2$ has values on $D_2$ domains: 1, \ldots, $D_2$, then the domains formed by $x_1$ and $x_2$ will be
a $D_1 \times D_2$ by 2 grid.

To get $100(1 - \alpha)\%$ approximate confidence intervals or surfaces for the domain means, we apply
the method in Meyer, M. C. (2018). $\hat{p}_J$ is the estimated probability that the projection of $y_s$ onto $C$
lands on $F_J$, and the $\hat{p}_J$ values are obtained by simulating many normal random vectors with
estimated domain means and covariance matrix $I$, where $I$ is a $M \times M$ matrix, and recording the
resulting sets $J$.

The user needs to provide a survey design, which is specified by the \texttt{svydesign} function in the
survey package, and also a data frame containing the response, predictor(s), domain variable, sampling weights, etc. So far, only stratified sampling design with simple random sampling without replacement (STSI) is considered in the examples in this package.

Note that when there might be any empty domain, the user must specify the total number of domains
in the \texttt{nD} argument.

For binomial and Poisson families use \texttt{family=quasibinomial()} and \texttt{family=quasipoisson()} to avoid a warning about non-integer numbers of successes. The ‘quasi’ versions of the family
objects give the same point estimates and standard errors and do not give the warning.

Value

The output is a list of values used for estimation, inference and visualization. Main output include:
survey.design The survey design used in the model.
etahat Estimated shape-constrained domain systematic component.
etahatu Estimated unconstrained domain systematic component.
muhat Estimated shape-constrained domain means.
muhatu Estimated unconstrained domain means.
lwr Approximate lower confidence band or surface for the shape-constrained domain mean estimate.
upp Approximate upper confidence band or surface for the shape-constrained domain mean estimate.
lwrnu Approximate lower confidence band or surface for the unconstrained domain mean estimate.
uppu Approximate upper confidence band or surface for the unconstrained domain mean estimate.
amat The $k \times M$ constraint matrix imposing shape constraints in each dimension, where $M$ is the total number of domains.
grid A $M \times p$ grid, where $p$ is the total number of predictors or dimensions.
nd A vector of sample sizes in all domains.
Ds A vector of the number of domains in each dimension.
acov Constrained mixture covariance estimate of domain means.
cov.un Unconstrained covariance estimate of domain means.
CIC The cone information criterion proposed in Meyer(2013a). It uses the "null expected degrees of freedom" as a measure of the complexity of the model. See Meyer(2013a) for further details of cic.
CIC.un The cone information criterion for the unconstrained estimator.
zeros_ps Index of empty domain(s).
nd Sample size of each domain.
pval p-value of the one-sided test.
family The family parameter defined in the formula.
df.residual The observed degree of freedom for the residuals of a csvy fit.
df.null The degree of freedom for the null model of a csvy fit.
domain Index of each domain in the data set contained in the survey.design object.
null.deviance The deviance for the null model of a csvy fit.
deviance The residual deviance of a csvy fit.
ans.unc_cp A data frame including the grid which is the combination of domains in each predictor, the domain mean estimate, and the constrained standard error.

Author(s)

Xiyue Liao
References

Oliva, C., Meyer, M. C., and Opsomer, J.D. (2020) Estimation and inference of domain means subject to qualitative constraints. *Survey Methodology*

See Also

`plotpersp`, to create a 3D Plot for a csvy Object with at least two predictors.
`incr`, to specify an increasing order constraint in a csvy formula.
`decr`, to specify a decreasing order constraint in a csvy formula.
`conc`, to specify a concave order constraint in a csvy formula.
`conv`, to specify a convex order constraint in a csvy formula.
`incr.conc`, to specify an increasing-concave order constraint in a csvy formula.
`decr.conv`, to specify a decreasing-convex order constraint in a csvy formula.
`decr.conc`, to specify a decreasing-concave order constraint in a csvy formula.
`incr.conv`, to specify an increasing-convex order constraint in a csvy formula.
`block.Ord`, to specify a blocking ordering order constraint in a csvy formula.
`svyby`, to compute survey statistics on subsets of a survey defined by factors.
`svymean`, to compute means for data from complex surveys.
`svyglm`, to fit a generalised linear model to data from a complex survey design, with inverse-probability weighting and design-based standard errors.

Examples

```r
# Example 1: monotonic in one dimension
data(api)
mcat <- api$meals
for(i in 1:10){mcat[trunc(api$meals/10) + 1 == i] <- i}
mcat[mcat == 100] <- 10
mcat <- factor(mcat)
M <- 10 # total number of domains

nsp<-c(200, 200, 200) ## sample sizes per stratum
es<-sample(api$sample[api$stype=='E' & !is.na(api$avg.ed) & !is.na(api$api00)],nsp[1])
ms<-sample(api$sample[api$stype=='M' & !is.na(api$avg.ed) & !is.na(api$api00)],nsp[2])
hs<-sample(api$sample[api$stype=='H' & !is.na(api$avg.ed) & !is.na(api$api00)],nsp[3])
```
sid<-c(es, ms, hs)

pw <- 1:6194*0 + 4421 / nsp[1]
pw[apipop$stype == 'M'] <- 1018 / nsp[2]
pw[apipop$stype == 'H'] <- 755 / nsp[3]

fpc <- 1:6194*0 + 4421
fpc[apipop$stype == 'M'] <- 1018
fpc[apipop$stype == 'H'] <- 755

strsamp <- cbind(apipop, mcat, pw, fpc)[sid,]
dstrat <- svydesign(ids = ~snum, strata = ~stype, fpc = ~fpc, data = strsamp, weight = ~pw)

rds <- as.svrepdesign(dstrat, type = "JKn")

anscl <- csvy(api00 ~ decr(mcat), design = rds, family = gaussian(), nd = M)
# summary(anscl)

# Example 2: unconstrained in x1 and increasing in x2 and x3

D1 <- 5
D2 <- 5
D3 <- 6
Ds <- c(D1, D2, D3)
M <- cumprod(Ds)[3] # total number of domains

x1vec <- 1:D1
x2vec <- 1:D2
x3vec <- 1:D3
grid <- expand.grid(x1vec, x2vec, x3vec)
N <- M*100*4
Ns <- rep(N/M, M)

mu.f <- function(x) {
  mus <- x[1]^0.25 + 4*exp(0.5 + 2*x[2]) / (1 + exp(0.5 + 2*x[2])) + sqrt(1/4 + x[3])
  mus <- as.numeric(mus$Var1)
  return (mus)
}

mus <- mu.f(grid)

H <- 4
nh <- c(180, 360, 360, 540)
n <- sum(nh)
Nh <- rep(N/H, H)

#generate population
y <- NULL
z <- NULL

for(i in 1:M){
  Ni <- Ns[i]
mui <- mus[i]

ei <- rnorm(Ni, 0, sd = 1)
yi <- mui + ei
y <- c(y, yi)
zi <- i/M + rnorm(Ni, mean = 0, sd = 1)
Z <- c(z, zi)

x1 <- rep(grid[,1], times = Ns)
x2 <- rep(grid[,2], times = Ns)
x3 <- rep(grid[,3], times = Ns)
domain <- rep(1:M, times = Ns)

c <- rep(1:N, times = Ns)
c <- c(c, cts)

c <- rep(1:N, times = Ns)
c <- c(c, cts)

c <- rep(1:N, times = Ns)
c <- c(c, cts)

c <- rep(1:N, times = Ns)
c <- c(c, cts)

pop <- data.frame(y = y, x1 = x1, x2 = x2, x3 = x3, domain = domain, strata = strata, w = w)
ssid <- stratsample(pop$strata, c("1" = nh[1], "2" = nh[2], "3" = nh[3], "4" = nh[4]))
sample.stsi <- pop[ssid, , drop = FALSE]
ds <- svydesign(id = ~1, strata = ~strata, fpc = ~freq, weights = ~w, data = sample.stsi)

#domain means are increasing w.r.t x1, x2 and block monotonic in x3
ord <- c(1, 1, 2, 2, 3, 3)
ans <- csvy(y ~ incr(x1)*incr(x2)*block.Ord(x3, order=ord), design = ds, family = gaussian(),
nD = M, test = FALSE, n.mix = 0)

#3D plot of estimated domain means: x1 and x2
plotpersp(ans)

#3D plot of estimated domain means: x3 and x2
plotpersp(ans, x3, x2)

#3D plot of estimated domain means: x3 and x2 for each domain of x1
plotpersp(ans, x3, x2, categ = "x1")

#3D plot of estimated domain means: x3 and x2 for each domain of x1
plotpersp(ans, x3, x2, categ = "x1", NCOL = 3)
Description

The NHANES study provides health data for a sample of the U.S. population. There are N=1680 observations with complete records for cholesterol level, age, height, waist size, and gender for adults ages 21-40.

Usage

data("nhdat")

Format

A data frame with 1680 observations on the following 7 variables.

id  ID vector
chol Cholesterol level: 1 (\geq 200 \text{ (mmol/L)}) and 0 (< 200 \text{ (mmol/L)})
wcat A factor of categorized waist size
gender Gender
age A factor of categorized age
wt A vector of stratified sampling weight
str A vector of strata variable in the stratified sampling design

Examples

library(csurvey)
data(nhdat)

#specify a stratified design
dstrat <- svydesign(ids = ~ id, strata = ~ str, data = nhdat, weight = ~ wt)

#constrained estimate: domain mean of cholesterol level is increasing in age and waist size
#M is the total number of domains
M <- 160
ans <- csvy(chol ~ incr(age) * incr(wcat) * gender, design = dstrat,
    nD = M, family = quasibinomial(link = "logit"), n.mix = 0, test = FALSE)

plotpersp(ans, categ = "gender", type = "response", NCOL = 2)
Index

* constrained regression
  csvy, 3
* datasets
  nhdat, 10
* shape routine
  block.Ord, 2

barplot.csvy (csvy), 3
barplot.svystat, 5
block.Ord, 2, 7

go.csvy (csvy), 3
conc, 7
destr.csvy (csvy), 3
conv, 7
csvy, 3

decr, 2, 7
decr.conc, 7
decr.conv, 7
deff.csvy (csvy), 3

fitted.csvy (csvy), 3
ftable.csvy (csvy), 3
ftable.svystat, 5

incr, 2, 7
incr.conc, 7
incr.conv, 7

nhdat, 10

plotpersp, 7
plotpersp.csvy (csvy), 3
predict.csvy (csvy), 3

SE.csvy (csvy), 3
summary.csvy (csvy), 3
svyby, 7
svycontrast, 5
svycontrast.csvy (csvy), 3

svyglm, 7
svymean, 7
vcov.csvy (csvy), 3