Package ‘csurvey’

May 22, 2022

Type Package

Title Constrained Regression for Survey Data

Version 1.3

Date 2022-05-21

Author Xiyue Liao

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

Description Domain mean estimation with monotonicity or block monotone constraints. See Xu X, Meyer MC and Opsomer JD (2021)<doi:10.1016/j.jspi.2021.02.004> for more details.

License GPL (>= 2)

Depends survey (>= 3.36), cgam (>= 1.7), R (>= 4.0)

Imports coneproj, purrr, stats, graphics, grDevices, MASS, Matrix

NeedsCompilation no

ByteCompile true

Repository CRAN

Date/Publication 2022-05-21 23:00:17 UTC

R topics documented:

`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

`csvy`

`csvy` *Estimation of Domain Means with Monotonicity or Convexity Constraints*

Description

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

For example, in a one dimensional situation, we assume that $\bar{y}_{U_t}$ 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 csvy function starts from the Hajek estimates $\bar{y}_{S_t} = (\sum_{k \in S_t} y_k/\pi_k)/N_t$ 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, $\hat{\boldsymbol{\theta}}$ is the minimizer of $(\tilde{\boldsymbol{y}}_S - \boldsymbol{\theta})^T \boldsymbol{W}_s (\tilde{\boldsymbol{y}}_S - \boldsymbol{\theta})$ subject to $\boldsymbol{A\theta} \geq \mathbf{0}$, where $\boldsymbol{W}_S$ is the diagonal matrix with elements $\hat{N}_1/\hat{N}, \ldots, \hat{N}_D/\hat{N}$, and $\hat{N} = \sum_{t=1}^T \hat{N}_t$ and $\boldsymbol{A}$ 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 $\mathcal{C}$ lands on $\mathcal{F}_{\mathcal{J}}$, 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 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 is any empty domain, the user must specify the total number of domains in the nD argument.

Usage

```
csvy(formula, design, subset=NULL, nD=NULL, family=stats::gaussian(),
    amat=NULL, level=0.95, n.mix=100L, test=TRUE,...)
## S3 method for class 'csvy'
summary(object,...)
## S3 method for class 'csvy'
vcov(object,...)
## S3 method for class 'csvy'
coef(object,...)

## S3 method for class 'csvy'
confint(object, parm=NULL, level = 0.95, type = c("link", "response"),...)
```

```
## S3 method for class 'csvy'
predict(object, newdata = NULL, type = c("link", "response"),
    se.fit = TRUE, level = 0.95, n.mix = 100,...)
```

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 convexity 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 μ is the vector of domain means and x is a predictor:

- incr(x): μ is increasing in x.
- decr(x): μ is decreasing in x.
- block.Ord(x): μ is has a block ordering 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 used in csvy: Gaussian, binomial, poisson, and Gamma.

`amat` A $k \times M$ matrix imposing shape constraints in each dimension, where M is the total number of domains. If the user doesn't provide the constraint matrix, a subroutine in the csurvey package will create a constraint matrix according to shape constraints specified in the formula. The default is amat = NULL.

`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.

`...` Other arguments

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.

`se.fit` Logical switch indicating if confidence intervals are required.

Details

For binomial and Poisson families use `family=quasibinomial()` and `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.

`predict` gives fitted values and sampling variability for specific new values of covariates. When `newdata` are the population mean it gives the regression estimator of the mean, and when `newdata` are the population totals and `total` is specified it gives the regression estimator of the population total. Regression estimators of mean and total can also be obtained with `calibrate`.

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.

`lwru` 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 a csvy 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.

Author(s)

Xiyue Liao

References

Xu, X. and Meyer, M. C. (2021) One-sided testing of population domain means in surveys.

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

Meyer, M. C. (2018) A Framework for Estimation and Inference in Generalized Additive Models with Shape and Order Restrictions. *Statistical Science* ***33(4)*** 595–614.

Wu, J., Opsomer, J.D., and Meyer, M. C. (2016) Survey estimation of domain means that respect natural orderings. *Canadian Journal of Statistics* ***44(4)*** 431–444.

Meyer, M. C. (2013a) Semi-parametric additive constrained regression. *Journal of Nonparametric Statistics* ***25(3)***, 715.

Lumley, T. (2004) Analysis of complex survey samples. *Journal of Statistical Software* ***9(1)*** 1–19.

See Also

`plotpersp`, to create a 3D Plot for a csvy Object

`incr`, to specify an increasing shape-restriction in a csvy Formula

`decr`, to specify an decreasing shape-restriction in a csvy Formula

Examples

```
data(api)

mcat = apipop$meals
for(i in 1:10){mcat[trunc(apipop$meals/10)+1==i] = i}
mcat[mcat==100]=10
D1 = 10

gcat = apipop$col.grad
for(i in 1:10){gcat[trunc(apipop$col.grad/10)+1==i] = i}
gcat[gcat >= 5] = 4
D2 = 4

nsp = c(200,200,200)  ## sample sizes per stratum

es = sample(apipop$snum[apipop$stype=='E'&!is.na(apipop$avg.ed)&!is.na(apipop$api00)],nsp[1])
ms = sample(apipop$snum[apipop$stype=='M'&!is.na(apipop$avg.ed)&!is.na(apipop$api00)],nsp[2])
hs = sample(apipop$snum[apipop$stype=='H'&!is.na(apipop$avg.ed)&!is.na(apipop$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,gcat,pw,fpc)[sid,]

dstrat = svydesign(ids=~snum, strata=~stype, fpc=~fpc, data=strsamp, weight=~pw)
rds = as.svrepdesign(dstrat, type=”JKn”)

# Example 1: monotonic in one dimension
ansc1 = csvy(api00~decr(mcat), design=rds, nD=D1)
# checked estimated domain means
# ansc1$muhat

# Example 2: monotonic in three dimensions
D1 = 5
D2 = 5
D3 = 6
Ds = c(D1, D2, D3)
M = cumprod(Ds)[3]

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

set.seed(1)
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)

cts = quantile(z, probs=seq(0,1,length=5))
strata = 1:N*0
strata[z >= cts[1] & z < cts[2]] = 1
strata[z >= cts[2] & z < cts[3]] = 2
strata[z >= cts[3] & z < cts[4]] = 3
strata[z >= cts[4] & z <= cts[5]] = 4
freq = rep(N/(length(cts)-1), n)

w0 = Nh/nh
w = 1:N*0
w[strata == 1] = w0[1]
w[strata == 2] = w0[2]
w[strata == 3] = w0[3]
w[strata == 4] = w0[4]
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, 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)

# Example 3: unconstrained in one dimension

#no constraint on x1
ans = csvy(y~x1*incr(x2)*incr(x3), design=ds, test=FALSE, n.mix=0)

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

Index