Package: SOR (via r-universe)

August 30, 2024

Type Package

Title Estimation using Sequential Offsetted Regression

Version 0.23.1

Date 2018-04-25

Depends Matrix

Imports methods, stats

Description Estimation for longitudinal data following outcome dependent sampling using the sequential offsetted regression technique. Includes support for binary, count, and continuous data. The first regression is a logistic regression, which uses a known ratio (the probability of being sampled given that the subject/observation was referred divided by the probability of being sampled given that the subject/observation was no referred) as an offset to estimate the probability of being referred given outcome and covariates. The second regression uses this estimated probability to calculate the mean population response given covariates.

License GPL-3

NeedsCompilation no

Author Lee McDaniel [aut, cre], Jonathan Schildcrout [aut]

Maintainer Lee McDaniel <lmcda4@lsuhsc.edu>

Date/Publication 2018-04-25 19:30:40 UTC

Repository https://lsubioscomputing.r-universe.dev

RemoteUrl https://github.com/cran/SOR

RemoteRef HEAD

RemoteSha 831027827b82faf379331a6a7bfedcdf530e2f06

Contents

sor *Sequentially Offsetted Regression*

Description

Fits model for data which was sampled based on a variable associated with the outcome. This function works for binary, count, and continuous responses.

Usage

```
sor(y.formula,
          w1.formula,
           w2.formula = ~1,
           id,
           waves = NULL,
           family = "binomial",
           y0 = 0,
           hfunc = identity,
           support = c(0,1),
           pi1.pi0.ratio = 1,
           data = parent.frame(),
           init.beta=NULL,
           init.sig.2 = 1,
           weights=NULL,
           est.var = TRUE,
           CORSTR="independence")
```

Arguments

y.formula	Regression formula for response
w1.formula	Formula for Z, not interacted with hfunc(Y). Of form Z~terms
w2.formula	Formula for Z, interacted with hfunc(Y). Of form ~terms
id	a vector identifying the clusters. By default, data are assumed to be sorted such that observations in a cluster are in consecutive rows and higher numbered rows in a cluster are assumed to be later. If NULL, then each observation is assigned its own cluster.
waves	an integer vector identifying components of a cluster. For example, this could be a time ordering. If integers are skipped within a cluster, then dummy rows with weight 0 are added in an attempt to preserve the correlation structure (except if corstr = "exchangeable" or "independent"). This can be skipped by setting nodummy=TRUE.
family	Character string representing reference distribution for the response. Can be one of "normal", "poisson", or "binomial".
y0	Representative value of response. Ignored if family="binomial".
hfunc	Function h, used with Y. Set to identity if family="binomial".

support	Values on which to evaluate the integrals. The lowest value should be less than the minimum response and the highest should be higher than the maximum response. If response is binary, support should be c(0,1). If response is count data, support should be an integer vector, for instance 0:50. If response is continuous, support should be a vector of points on which to integrate.
pi1.pi0.ratio	The referral ratio
data	Data frame or environment with all the data
init.beta	Initial values for parameters in y.formula. Convergence may depend heavily on the initial values used. If family="binomial", the default is recommended.
init.sig.2	Initial value for sigma^2. Only for family="normal".
weights	A vector of weights for each observation. If an observation has weight 0, it is excluded from the calculations of any parameters. Observations with a NA anywhere (even in variables not included in the model) will be assigned a weight of 0. This should normally be used to preserve the correlation structure.
est.var	Logical. Should the variance be estimated. Only for family="normal".
CORSTR	Correlation structure

Value

Returns a list with values from the fit.

Author(s)

Lee S. McDaniel, Jonathan S. Schildcrout

References

This package relies heavily on code from geeM:

McDaniel, L. S., Henderson, N. C., & Rathouz, P. J. (2013). Fast pure R implementation of GEE: application of the matrix package. The R journal, 5(1), 181.

Examples

```
generatedata <- function(beta,alpha,X,ntime,nsubj, betat, betat1)  {

  mean.vec <- exp(crossprod(t(X), beta))
  y <- matrix(0,nrow=nsubj, ncol=ntime)
  y[,1] <- rpois(nsubj ,lambda = mean.vec)
  old.mean <- mean.vec
  new.mean <- old.mean*exp(betat + betat1*X[,2])
  for (t in 1:(ntime-1))  {
    lambda.t <- new.mean - alpha*sqrt(old.mean*new.mean)
    theta.t <- alpha*sqrt(new.mean/old.mean)
    I <- rpois(nsubj, lambda = lambda.t)
    W <- rbinom(nsubj, y[,t], theta.t)

    y[,t+1] = W + I
    old.mean <- new.mean
```

```
    new.mean <- old.mean*exp(betat + betat1*X[,2])
  }
  longform <- c(t(y))
  time <- rep(1:ntime,times=nsubj)
  subject <- rep(c(1:nsubj),each=ntime)

  simdata <- data.frame(count = longform, time = time, subject=subject)
  return(simdata)
}
logit <- function(p) log(p)-log(1-p)
expit <- function(x) exp(x)/(1+exp(x))
set.seed(1)

npop <- 10000
beta0 <- -1.4
beta1 <- 0.4
alpha <- 0.9
gam0 <- -3.15
gam1 <- 6.3
nsubj <- 200
ntime <- 8
betat <- -0.1; betat1 <- 0.1
thresh <- 1

x0 <- rep(1, npop)
x1 <- rbinom(npop, 1, 0.5)

Xmat <- cbind(x0, x1)
timevec <- 0:(ntime-1)

testdat <- generatedata(c(beta0, beta1), alpha, Xmat, ntime, npop, betat = betat, betat1 = betat1)
Y <- matrix(testdat$count, nrow=npop, ncol=ntime, byrow=TRUE)
lambdap <- expit(gam0 + gam1*as.numeric(Y[,1]>=thresh))
Z <- rbinom(npop, 1, lambdap)

casesamp <- rep(0, npop)
casesamp[Z==1] <- rbinom(sum(Z), 1, nsubj/(2*sum(Z)))
controlsamp <- rep(0, npop)
controlsamp[Z==0] <- rbinom(sum(1-Z), 1, nsubj/(2*sum(1-Z)))


case <- which(casesamp==1)
control <- which(controlsamp==1)
id <- sort(c(case, control))
nsubj <- length(control) + length(case)
Ysamp <- NULL
lamsamp <- NULL
zsamp <- NULL
x1samp <- NULL
idsamp <- NULL
time <- NULL
obspersubj <- sample(3:ntime, size=nsubj, replace=TRUE)
for(i in 1:nsubj){
```

```
  Ysamp <- c(Ysamp, Y[id[i],1:obspersubj[i]])
  zsamp <- c(zsamp, rep(as.numeric(Z[id[i]]), obspersubj[i]))
  x1samp <- c(x1samp, rep(x1[id[i]], obspersubj[i]))
  time <- c(time, 0:(obspersubj[i]-1))
  idsamp <- c(idsamp, rep(i, obspersubj[i]))
}
p1p0 <- sum((1-Z))/sum(Z)

timemax <- pmax(time-2, 0)
y0 <- 1
betas <- c(beta0, beta1, betat, betat1)
init <-  runif(4, betas -0.1, betas + 0.1)

y.formula <- y~x1+time + x1:time
w1 <- z ~ x1+ as.factor(time) +  x1:time + x1:timemax
w2 <- ~x1 + time + timemax + x1:time + x1:timemax

DAT.ods <- data.frame("x1"= x1samp, "time" = time,
                      "timemax" = timemax, "z" = zsamp, "y" = Ysamp, "id" = idsamp)

sor(y.formula, w1, w2, id, family="poisson",y0=1,
    support=0:25, pi1.pi0.ratio=p1p0, data=DAT.ods, init.beta=init, CORSTR="ar1")
```

Index