cutpoints

Choose Cutpoints for Categorizing a Continuous Predictor

DESCRIPTION:

Choose the number and location of cutpoints for categorizing a continuous predictor variable

USAGE

cutpoints(mean.vector, variance.vector=rep(1,length(mean.vector)),
frequency.variable, k, y.data, KMAX=20, scale.parameter=1,
x.data=1:length(mean.vector), exact=T, equal.size =F, aed.table=T,
cutpoints.matrix=F, return.factor=T, use.C=F)

REQUIRED ARGUMENTS

mean.vector .
An estimate of the statistical relationship \(E[y|x = x.data]\), where x is the predictor variable, y is the outcome variable, and x.data and y.data are vectors of the observed values of x and y in the dataset. NOTE: This estimate can be obtained via nonparametric regression. See notes below and S-Plus documentation for loess(), smooth.spline(), supsmu(), gam().

OPTIONAL ARGUMENTS

variance.vector (numeric vector)
The variance of the outcome variable (assumed to be a known function of the mean).

frequency.vector (integer vector)
Each element gives the frequency of occurrence of the corresponding element of the input vectors (mean.vector, variance.vector, y.data, and x.data). If missing, each element is assumed to occur 1 time. If exact = F, then frequency.vector must be missing, since the approximate method assumes no ties in x.data.

k (integer)
The number of categories to make. If missing, cutpoints will choose the number of categories by minimizing the objective function over all choices of k from 1, . . . , KMAX.

y.vector (numeric vector)
The value of the outcome variable. Not used in computations for choosing cutpoints. If supplied, the sum, average and standard error of the outcome variable in each category is displayed in a table.
KMAX (integer)
The maximum number of categories to try when searching for the optimum number of
categories. Cutpoints tries all possible possible values in 1, . . . , KMAX, where KMAX = 20 by default.

scale.parameter (numeric)
An estimate of the scale parameter. By default, the scale parameter is taken to be 1.

x.vector (numeric vector)
The value of the predictor variable for each observation. If missing, y.vector is assumed
to be sorted by increasing values of the predictor variable and category cutpoints are
identified in terms of indexes.

exact (logical)
If true, the number and location of cutpoints are chosen to make the objective function
a global minimum. If false, an approximate solution is obtained.

equal.size logical value: If true, the categories are restricted to contain an approximately
equal number of observations, and the objective function is minimized with respect to
the number of equal size categories.

aed.table logical value: If true, the returned object includes a table containing the mini-
mum possible value of the objective function for each fixed number of categories k = 1, . . . , KMAX.

cutpoints.matrix (logical)
If true, the returned object includes the best cutpoint locations for each fixed number
of categories k = 1, . . . , KMAX.

return.factor (logical)
If true, the returned object includes a factor object of length(mean.vector) that indi-
cates the category to which each observation is assigned.

use.C (logical)
If true, call a C function to perform the computations. This option is not currently
available for distribution. Use the default option, use.C = F.

VALUE:

AED
A table showing the minimum value of the objective function for each number of
categories k = 1, . , KMAX. This component is suppressed If the argument k is used.

k
The number of categories. aed The value of the objective function for the selected
categorization.
cutpoints.x

The vector of cutpoint locations. If x.vector is not supplied, then cutpoint location are specified by the number of observations in each category. TABLE A summary table showing the number of observations in each category (ignoring weights). If y.vector is supplied, the summary table also shows the sum and average of the outcome variable in each category.

DETAILS:

The cutpoints() function and other utility functions are distributed in a file called CUTPOINT.FUNCTIONS.s. The user should use: source("CUTPOINT.FUNCTIONS.s") to read these functions into S-Plus. Utility functions defined in CUTPOINT.FUNCTIONS.s are: cp.p(), aed.p(), cat.table(), aed.quantiles(), aed.approx(), aed.exact(), and pwc.sse().

The goal of cutpoints() is to determine the number and location of category cutpoints in order to minimize information loss and maximize precision when estimating a regression function. The number and location of category cutpoints are chosen to minimize the estimated average expected distance (AED) objective function, as defined in O'Brien (2004a). The general strategy is: (1) estimate the unknown regression function via smoothing; and (2) use the smooth estimate of the regression function to identify good cutpoints. The cutpoints() function performs part (2) only. It is up to the user to perform step (1) before calling cutpoints(). S-Plus has several built-in smoothing functions; sample S-Plus code is provided below. It is recommended that a rank transformation be applied to the predictor variable before estimating the regression function in step (1).

If exact = T, cutpoints() chooses the number and location of cutpoints to correspond to an exact minimum of the estimated AED. Despite using an efficient algorithm (O'Brien, 2004b), this approach is computationally intensive and may be infeasible for large datasets. If exact = F, a fast approximate solution is obtained using the asymptotic approach of O'Brien (2004a). The approximate method assumes no ties in the predictor variable. The current implementation differs from O'Brien (2004a) in some respects. Unlike O'Brien (2004a), the number of categories is chosen by searching over 1,2,...,KMAX, where KMAX = 20 by default.

The objective function that cutpoints() minimizes has the form:

\[
\text{AED} = \frac{1}{n} \sum_{j=1}^{k} \sum_{i=t_{j-1}+1}^{t_j} \frac{(\bar{\mu}_j - \mu_i)^2 + \sigma^2 \bar{v}_j / n_j}{v_i}
\]

where \( t_0 = 0, t_k = n; k \) is the number of categories; \( n \) is the sample size; \( \{t_j\} \) are indexes corresponding to cutpoint locations; \( \mu_i \) is an estimate of the expected value of the outcome variable for the \( i \)th subject; \( v_i = V(\mu_i) \) is the estimated variance function; \( \sigma^2 \) is an estimate of the scale parameter; \( \bar{\mu}_j = n_j^{-1} \sum_{i=t_{j-1}+1}^{t_j} \mu_i; \bar{v}_j = n_j^{-1} \sum_{i=t_{j-1}+1}^{t_j} v_i; n_j = t_j - t_{j-1}; \) and the data are assumed to be sorted such that the \( i \)th subject is the one with the \( i \)th smallest predictor
value. The minimization is performed with respect to the choice of $k$ and $\{t_j\}$. If exact = T, the minimization is exact. If exact = F, an approximate minimization is obtained using the method of O’Brien (2004a).

REFERENCES:


EXAMPLES:

############################################################
########### EXAMPLE 1 ##############
############################################################

### hiv.status is a Bernoulli (0/1) outcome variable
### age is a continuous predictor variable with no ties.

### Apply rank transformation to age ###
age.rank <- rank(age)
### Estimate the age-hiv relationship using smoothing splines###
smooth.fit <- gam(hiv.status ~ s(age.rank),family=binomial)
### Extract fitted values ###
fitted.vals <- fitted(smooth.fit)
### Compute the Bernoulli variance ###
var.func <- fitted.vals*(1-fitted.vals)
### Choose cutpoints ###
result <- cutpoints(fitted.vals, var.func, y=hiv.status, x=age, exact = F)
### Print result
print(result)

############################################################
########### EXAMPLE 2 ##############
############################################################

### fev is a continuous outcome variable
### age is a numeric predictor variable possibly with ties

###Apply rank transformation to age ###
age.rank <-rank(age)
### Estimate the age-fev relationship using loess ###
smooth.fit <- gam(fev~ s(age.rank),family=gaussian)
### Extract fitted values ###
fitted.vals <- fitted(smooth.fit)
### Extract the estimated variance ###
dispersion <- summary(smooth.fit)$dispersion
### choose cutpoints ###
result <- cutpoints(fitted.vals, y=fev, x=age, scale = dispersion, exact = T)
### Print result ###
print(result)

#####################################################
####### CODE FOR OTHER SMOOTHING FUNCTIONS #######
#####################################################

#####################################################
### smooth.spline: smoothing splines ###
#####################################################
### Use df = to control the amount of smoothing
smooth.spline(x,y,df=4)
### Use cv = T to choose the smoothing parameter automatically
smooth.spline(x,y, cv = T)

#####################################################
### loess: local polynomial regression ###
#####################################################
### Use enp.target= or span= to control the amount of smoothing
### Use degree = to specify local linear or local quadratic regression
loess(x,y, degree=1, enp.target = 4)
loess(x,y, degree=2, span = 0.75)