Package ‘StagedChoiceSplineMix’

August 11, 2016

Type Package

Title Mixture of Two-Stage Logistic Regressions with Fixed Candidate Knots

Version 1.0.0

Date 2016-08-10

Author Elizabeth Bruch <ebruch@umich.edu>, Fred Feinberg <feinf@umich.edu>, Kee Yeun Lee <keeyeun.lee@polyu.edu.hk>

Maintainer Kee Yeun Lee <keeyeun.lee@polyu.edu.hk>

Description Analyzing a mixture of two-stage logistic regressions with fixed candidate knots. See Bruch, E., F. Feinberg, K. Lee (in press)<DOI:10.1073/pnas.1522494113>.

Depends R (>= 3.0.1), plyr, base, stats, methods

License GPL-2

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-08-11 13:13:59

R topics documented:

bs.se .................................................. 2
gen.init .............................................. 2
move.knot .......................................... 3
StagedChoiceSplineMix ............................ 4
twostglogitregmixEM .............................. 9

Index 11
### bs.se

**Performs a parametric bootstrapping standard error approximation**

#### Description

The function performs a parametric bootstrapping standard error approximation for mixtures of two-stage logistic regressions.

#### Usage

```
bs.se(output, B = 100)
```

#### Arguments

- **output**: output of `StagedChoiceSplineMix`  
- **B**: number of bootstrap samples (def:100)

#### See Also

- `StagedChoiceSplineMix`  
- `boot.se`

#### Examples

```r
## parametric bootstrapping to calculate standard error:  
## use best output from StagedChoiceSplineMix exampl
#out.se<-bs.se(output=out$best, B=100)
#out.se$betab.se  
#out.se$betaw.se  
#out.se$lambda.se
```

### gen.init

**Generates initial values for mixtures of logistic regressions**

#### Description

A sub-function of `StagedChoiceSplineMix`. This function generates initial values for mixtures of logistic regressions. This function is used if starting points for parameters are not specified by the user or when the EM algorithm needs to be initialized due to errors.

#### Usage

```
gen.init(y, x, k, er)
```
**move.knot**

**Arguments**

- `y` See `StagedChoiceSplineMix` for details.
- `x` See `StagedChoiceSplineMix` for details.
- `k` See `StagedChoiceSplineMix` for details.
- `er` The total number of errors. See `StagedChoiceSplineMix` for details.

**See Also**

`StagedChoiceSplineMix`

"mixtools" package version 1.0.3

---

**move.knot**  
*Generates a new set of knots for the following iteration*

---

**Description**

A sub-function of `StagedChoiceSplineMix`. This function generates a new set of knots for the following iteration. Please refer to Bruch et al. (in press) for the precise rule used.

**Usage**

```r
move.knot(num.knot, sp.knots, k)
```

**Arguments**

- `num.knot` See `StagedChoiceSplineMix` for details.
- `sp.knots` See `StagedChoiceSplineMix` for details.
- `k` See `StagedChoiceSplineMix` for details.

**References**


**See Also**

`StagedChoiceSplineMix`
StagedChoiceSplineMix  Performs iterations between an EM algorithm for a mixture of two-stage logistic regressions with fixed candidate knots and knot movements

Description

The function performs iterations between an EM algorithm for a mixture of two-stage logistic regressions with fixed candidate knots and knot movements. The function generates candidate knots for each splined variable. Three sub-functions (gen.init, twostglogitregmixEM, move.knot) are used within the function.

Usage

StagedChoiceSplineMix(data = NULL, M = 100, sp.cols = NULL, num.knots = NULL, sp.knots = NULL, betab = NULL, betaw = NULL, lambda = NULL, k = 2, nst = 20, epsilon = 1e-06, maxit = 500, maxrestarts = 100, maxer = 20)

Arguments

data  Raw data for StagedChoiceSplineMix

Format

• 1st column: id
• 2nd column: the 1st stage binary variable (browsing)
• 3rd column: the 2nd stage binary variable (writing) conditional on the 1st stage binary variable. It should be left blank (or NA) if 1st stage variable is equal to 0
• The rest of columns: covariates including splined variables

See Format (below) for details

M  number of iterations (def: 100)

sp.cols  vector of column numbers of splined variables in a data set (if sp.col is 0, twostglogitregmixEM function should be used)

num.knots  vector of numbers of knot candidates for splined variables. (def: a vector all of whose entries are "19")

sp.knots  list of knot configurations. For each splined variable, a knot configuration is a k by 4 matrix whose rows represent latent classes and columns represent knots [browsing knot 1, browsing knot 2, writing knot 1, writing knot 2]. (def: approximately 1/3 and 2/3 of knot candidates for knot 1 and knot 2 respectively)

betab  matrix of starting points for betab (browsing parameters). If not given, gen.init generates starting points.

betaw  matrix of starting points for betaw (writing parameters). If not given, gen.init generates starting points.
lambda vector of starting points for lambda (membership proportion). If not given, `gen.init` generates starting points

k number of latent classes (def: 2)

nst number of random multiple starting points to try given a knot configuration. For each knot configuration, the output with the largest log-likelihood is stored among `nst` trials. (def: 20)

epsilon stopping tolerance for the EM algorithm. (def: 1e-06)

maxit maximum number of the EM iterations allowed. If convergence is not declared before `maxit`, the EM algorithm stops with an error message and generates new starting points. (def: 500)

maxrestarts maximum number of restarts (due to a singularity problem) allowed in the EM iterations. If convergence is not declared before `maxrestarts`, the algorithm stops with an error message and generates new starting points. (def: 100)

maxer maximum number of errors allowed within a given knot configuration. If convergence is not declared before `maxer`, it tries a new knot configuration. (def: 20)

Format

The simulated data (`simdata`) in the examples is generated using information below:

1. **User identifier**: `userid`
   - Number of users: 700
   - Number of observations per user: 200

2. **Dependent variables**:
   - `browsed`: 1st stage binary dependent variable
   - `wrote`: 2nd stage binary dependent variable

3. **Covariates**:
   - `x1` (discrete variable): random draws from a binomial distribution with 0.7 success probability
   - `sp1` (splined variable): random draws from a uniform distribution between -2 and 2

4. **Number of latent classes**: 3

5. **True parameters**:

   i.Betab(browsing)

<table>
<thead>
<tr>
<th></th>
<th>Class1</th>
<th>Class2</th>
<th>Class3</th>
</tr>
</thead>
<tbody>
<tr>
<td>intercept</td>
<td>1.5</td>
<td>2</td>
<td>0.3</td>
</tr>
<tr>
<td>x1</td>
<td>-0.2</td>
<td>-0.1</td>
<td>0.6</td>
</tr>
<tr>
<td>sp1</td>
<td>-2.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>sp1 knot1</td>
<td>3</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>sp1 knot2</td>
<td>2</td>
<td>-0.5</td>
<td>-1.5</td>
</tr>
</tbody>
</table>
ii. Betaw(writing)

<table>
<thead>
<tr>
<th></th>
<th>Class1</th>
<th>Class2</th>
<th>Class3</th>
</tr>
</thead>
<tbody>
<tr>
<td>intercept</td>
<td>-2</td>
<td>-1</td>
<td>0.3</td>
</tr>
<tr>
<td>x1</td>
<td>-0.3</td>
<td>-0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>sp1</td>
<td>-3</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>sp1 knot1</td>
<td>3</td>
<td>2</td>
<td>-1.5</td>
</tr>
<tr>
<td>sp1 knot2</td>
<td>3</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

iii. Lambda (membership proportion)

<table>
<thead>
<tr>
<th></th>
<th>Class1</th>
<th>Class2</th>
<th>Class3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

6. True knot configuration:
19 candidate knots for sp1 (20-iles)

<table>
<thead>
<tr>
<th>knot1 (browsing)</th>
<th>knot2 (browsing)</th>
<th>knot1 (writing)</th>
<th>knot2 (writing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class1</td>
<td>8</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Class2</td>
<td>5</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Class3</td>
<td>5</td>
<td>13</td>
<td>7</td>
</tr>
</tbody>
</table>

Value

`StagedChoiceSplineMix` returns a list of the following items:

- best: best output, i.e., that giving the largest log-likelihood among M outputs.
  - best$loglik$: log-likelihood
  - best$betab$: parameter estimates of betab
  - best$betaw$: parameter estimates of betaw
  - best$lambda$: parameter estimates of lambda
  - best$sp.knots$: knot configuration

- loglike: vector of log-likelihoods for M outputs.

References


See Also

- `gen.init.move.knot twostglogitregmixEM`
- "mixtools" package version 1.0.3
### Examples

#### 1. Generate data (simdata)

```r
set.seed(77)
k <- 3

beta_b <- matrix(c(1.5, 2.0, 0.3, -0.2, -0.1, 0.6, -2.5, 0.5, 1.3, -1.1, 1.2, -0.5, -1.5), 5, k, byrow=TRUE)
beta_w <- matrix(c(-2.0, -1.0, 3.0, -0.3, -0.2, 0.2, -3.2, 3.2, -1.5, 3.2, -1), 5, k, byrow=TRUE)
spl1.knots <- matrix(c(8, 14, 4, 11, 5, 15, 5, 12, 5, 13, 7, 14), 3, 4, byrow=TRUE)
lambda <- c(0.3, 0.3, 0.4)

# Large data set
n_id <- 700
nobsb <- 200

# Small data set
n_id <- 100
nobsb <- 100

nb <- n_id * nobsb
idb <- rep(1:n_id, each=nobsb)
xb.1 <- rbinom(nb, size=1, prob=0.7)
xb.com <- cbind(1, xb.1)
xb.sp1 <- runif((nb), -2, 2)
xb <- cbind(xb.com, xb.sp1)

sp1.mat.b <- matrix(double(20*nb), ncol=20)
sp1.mat.b[,1] <- xb.sp1
sp1.quan <- quantile(xb.sp1, c(0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95))

for (i in 1:19){
  sp1.mat.b[, (i+1)] <- xb.sp1 - sp1.quan[i]
}
sp1.mat.b[sp1.mat.b<0] <- 0
sp1.mat.b[,1] <- xb.sp1

xb.class <- list()
for (i in 1:k){
  xb.class[[i]] <- cbind(xb.com, sp1.mat.b[,1], sp1.mat.b[, (sp1.knots[i, 1:2]) + 1])
}

xbetab <- matrix(double(nb*k), ncol=k)
for (i in 1:k){
  xbetab[,i] <- data.matrix(xb.class[[i]]) %*% xbetab[,i]
}

num.idb <- as.vector(ddply(data.frame(idb), "idb", count),[-1])

w <- sample(c(1:k), n_id, replace = TRUE, prob = lambda)
```
StagedChoiceSplineMix

```R
wb <- rep(w, num.idb)
yb.temp <- matrix(double(nb*k), ncol = k)

for (i in 1:k){
yb.temp[, i] <- rbinom((nb), size = 1, prob = (1/(1+exp(-xbetab[, i]))))
}
yb <- sapply(1:nb, function(i) yb.temp[i, wb[i]])

idw <- idb[yb == 1]
nw <- length(idw)
spl.mat.w <- spl.mat.b[yb == 1, ]
xw.com <- xb.com[yb == 1, ]
xw <- xb[yb == 1, ]

xw.class <- list()
for (i in 1:k){
xw.class[[i]] <- cbind(xw.com, spl.mat.w[, 1], spl.mat.w[, (spl.knots[i, 3:4]) + 1])
}

xbetaw <- matrix(double(nw*k), ncol = k)
for (i in 1:k){
  xbetaw[, i] <- data.matrix(xw.class[[i]]) %*% xbetaw[, i]
}

num.idw <- as.vector(ddply(data.frame(idw), "idw", count)[,-1])
ww <- wb[yb == 1]

yw.temp <- matrix(double(nw*k), ncol = k)
for (i in 1:k){
yw.temp[, i] <- rbinom((nw), size = 1, prob = (1/(1+exp(-xbetaw[, i]))))
}
yw <- sapply(1:nw, function(i) yw.temp[i, ww[i]])

yb.aug <- cbind(1: length(yb), yb)
yw.aug <- cbind(which(yb == 1), yw)

colnames(yb.aug)[1] <- "num"
colnames(yw.aug)[1] <- "num"

ybyw <- merge(yb.aug, yw.aug, all = TRUE)[,-1]
simdata <- cbind(idb, ybyw, xb.1, spl.mat.b[, 1])
simdata[is.na(simdata)] <- ""
simdata[, 3] <- as.integer(simdata[, 3])
colnames(simdata)[1] <- "userid"
colnames(simdata)[2] <- "browsed"
colnames(simdata)[3] <- "wrote"
colnames(simdata)[4] <- "x1"
colnames(simdata)[5] <- "sp1"

########################################################################
#### 2. Run StagedChoiceSplineMix ####
## number of latent classes
```

## twostglogitregmixEM

Performs an EM algorithm for mixtures of two-stage logistic regressions

### Description

A sub-function of `StagedChoiceSplineMix`. This function performs an EM algorithm for mixtures of two-stage logistic regressions.

### Usage

```r
twostglogitregmixEM(yb = NULL, idb = NULL, yw = NULL, idw = NULL, xb = NULL, xw = NULL, xb.class = NULL, xw.class = NULL, sp.cols = NULL, num.knots = NULL, sp.knots = NULL, betab = NULL, betaw = NULL, lambda = NULL, k = 3, nst = 1, epsilon = 1e-06, maxit = 100, maxrestarts = 100, maxer = 20)
```

**set.seed(66)**

```r
k<-3
```

```r
## starting points: true parameters used in the data generation (optional)
betab<-matrix(c(1.5,2.0,0.3,-0.2,-0.1,0.6,-2.5,0.5,1,3,-1,1,2,-0.5,-1.5),5,k,byrow=TRUE)
betaw<-matrix(c(-2.0,-1,0.3,-0.3,-0.2,0.2,-3,-2,0,3,2,-1.5,3,2,-1),5,k,byrow=TRUE)
lambda<-c(0.3,0.3,0.4)
```

```r
## number of random multiple starting points to try given a knot configuration
nst<-1
```

```r
## vector of the columns of spline variables in the data set (required)
sp.cols<-5
```

```r
# vector of the numbers of candidate knots for splined variables (optional)
um.knots<-19
```

```r
## true knot configuration used in the data generation
sp1.knots<-matrix(c(8,14,4,11,5,15,5,12,5,13,7,14),3,4,byrow=TRUE)
```

```r
## list of knot configuration of spline variables (optional)
sp.knots<-list(sp1.knots)
```

## Run "StagedChoiceSplineMix"

```r
out<StagedChoiceSplineMix(data=simdata, M=1, sp.cols=sp.cols, num.knots, sp.knots, betab, betaw, lambda, k=k, nst=nst, epsilon=1e-06, maxit=500, maxrestarts=100, maxer=20)
```

```r
## output
out$loglike # vector of M log-likelihoods
out$best$loglik # log-likelihood of the best output
out$best$betab # betab estimates of the best output
out$best$betaw # betaw estimates of the best output
out$best$lambda # lambda estimates of the best output
out$best$sp.knots # knot configuration of the best output
```
betaw = NULL, lambda = NULL, k = NULL, epsilon = 1e-06, maxit = 500, 
maxrestarts = 100, maxer = 20, verb = FALSE)

Arguments

yb The 1st stage binary variable (browsing). See StagedChoiceSplineMix for details.
idb Corresponding id for yb. See StagedChoiceSplineMix for details.
yw The 2nd stage binary variable (writing). See StagedChoiceSplineMix for details.
idw Corresponding id for yw. See StagedChoiceSplineMix for details.
xb Corresponding covariance matrix for yb. See StagedChoiceSplineMix for details.
xw Corresponding covariance matrix for yw. See StagedChoiceSplineMix for details.
xb.class Corresponding latent classes for yb.
xw.class Corresponding latent classes for yw.
sp.cols See StagedChoiceSplineMix for details.
num.knots See StagedChoiceSplineMix for details.
sp.knots See StagedChoiceSplineMix for details.
betab See StagedChoiceSplineMix for details.
betaw See StagedChoiceSplineMix for details.
lambda See StagedChoiceSplineMix for details.
k See StagedChoiceSplineMix for details.
epsilon See StagedChoiceSplineMix for details.
maxit See StagedChoiceSplineMix for details.
maxrestarts See StagedChoiceSplineMix for details.
maxer See StagedChoiceSplineMix for details.
verb See StagedChoiceSplineMix for details.

See Also

StagedChoiceSplineMix
regmixEM
Index

boot.se, 2
bs.se, 2

gen.init, 2, 4–6
move.knot, 3, 4, 6
regmixEM, 10
StagedChoiceSplineMix, 2, 3, 4, 9, 10
twostglogitregmixEM, 4, 6, 9