

# Package ‘AIM’

February 14, 2012

**Title** AIM: adaptive index model

**Version** 1.01

**Author** L. Tian and R. Tibshirani

**Description** R functions for adaptively constructing index models for continuous, binary and survival outcomes. Implementation requires loading R-pacakge “survival”

**Maintainer** Robert Tibshirani <tibs@stat.stanford.edu>

**LazyLoad** false

**LazyData** false

**Depends** survival

**Imports** survival

**License** LGPL-2

**Repository** CRAN

**Date/Publication** 2010-04-05 19:01:20

## R topics documented:

backfit.cox.interaction . . . . .	2
backfit.cox.main . . . . .	2
backfit.lm.interaction . . . . .	2
backfit.lm.main . . . . .	3
backfit.logistic.interaction . . . . .	3
backfit.logistic.main . . . . .	3
cox.interaction . . . . .	4
cox.main . . . . .	5
cv.cox.interaction . . . . .	7
cv.cox.main . . . . .	9
cv.lm.interaction . . . . .	11

cv.lm.main . . . . .	13
cv.logistic.interaction . . . . .	15
cv.logistic.main . . . . .	17
index.prediction . . . . .	19
lm.interaction . . . . .	20
lm.main . . . . .	22
logistic.interaction . . . . .	23
logistic.main . . . . .	25

**Index** **28**

backfit.cox.interaction

*Internal function used in cox.interaction*

**Description**

Internal functions used in cox.interaction

**Author(s)**

Lu Tian and Robert Tibshirani

backfit.cox.main

*Internal function used in cox.main*

**Description**

Internal functions used in cox.main

**Author(s)**

Lu Tian and Robert Tibshirani

backfit.lm.interaction

*Internal function used in lm.interaction*

**Description**

Internal functions used in lm.interaction

**Author(s)**

Lu Tian and Robert Tibshirani

---

`backfit.lm.main`      *Internal function used in lm.main*

---

**Description**

Internal functions used in lm.main

**Author(s)**

Lu Tian and Robert Tibshirani

---

`backfit.logistic.interaction`  
*Internal function used in logistic.interaction*

---

**Description**

Internal functions used in logistic.interaction

**Author(s)**

Lu Tian and Robert Tibshirani

---

`backfit.logistic.main`      *Internal function used in logistic.main*

---

**Description**

Internal functions used in logistic.main

**Author(s)**

Lu Tian and Robert Tibshirani

---

cox.interaction	<i>Interaction Cox adaptive index model</i>
-----------------	---

---

### Description

Estimate adaptive index model for survival outcomes in the context of Cox regression. The resulting index characterizes the interaction between covariates and treatment.

### Usage

```
cox.interaction(x, trt, y, delta, nsteps=8, mincut=.1, backfit=F, maxnumcut=1, dirp=0)
```

### Arguments

x	n by p matrix. The covariate matrix
trt	n vector. The treatment indicator
y	n vector. The observed follow-up time
delta	n 0/1 vector. The status indicator. 1=failure and 0=alive.
nsteps	the maximum number of binary rules to be included in the index
mincut	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
backfit	T/F. Whether the existing split points are re-adjusted after including new a binary rule
maxnumcut	the maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)" and -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

### Details

cox.interaction sequentially estimates a sequence of adaptive index models with up to "nsteps" terms for survival outcomes. The algorithm seeks the index having the strong interaction with the treatment in the Cox regression. The appropriate number of binary rules can be selected via K-fold cross-validation (cv.cox.interaction).

### Value

cox.interaction returns maxsc, which is the observed partial likelihood score test statistics for the index\*treatment interaction in the fitted model and res, which is a list with components

jmaa	number of predictors
cutp	split points for the binary rules
maxdir	direction of split: 1 represents "(x>cut)" and -1 represents "(x<cut)"
maxsc	observed partial likelihood score test statistics for the interaction

**Author(s)**

Lu Tian and Robert Tibshirani

**References**

Lu Tian and Robert Tibshirani (2010) "Adaptive index models for marker-based risk stratification", Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

**Examples**

```
## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
trt=rbinom(n,1,0.5)
fail.time=rexp(n)*exp(-beta*z*trt)
cen.time=rexp(n)*1.25
y=pmin(fail.time, cen.time)
y=round(y*10)/10
delta=1*(fail.time<cen.time)

## fit the interaction Cox AIM model
a=cox.interaction(x, trt, y, delta, nsteps=10)

## examine the model sequence
print(a)

## compute the index based on the 2nd model in the sequence using data x
z.prd=index.prediction(a$res[[2]],x)

## compute the index based on the 2nd model of the sequence using new data xx, and compare the result with the true index
nn=10
xx=matrix(rnorm(nn*p), nn, p)
zz=(xx[,1]<0.2)+(xx[,5]>0.2)
zz.prd=index.prediction(a$res[[2]],xx)
cbind(zz, zz.prd)
```

---

cox.main

*Main effect Cox adaptive index model*

---

**Description**

Estimate adaptive index model for survival outcomes in the context of Cox regression. The resulting index characterizes the main covariate effect on the hazard.

**Usage**

```
cox.main(x, y, delta, nsteps=8, mincut=.1, backfit=F, maxnumcut=1, dirp=0)
```

**Arguments**

x	n by p matrix. The covariate matrix
y	n vector. The observed follow-up time
delta	n 0/1 vector. The status indicator. 1=failure and 0=alive.
nsteps	the maximum number of binary rules to be included in the index
mincut	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
backfit	T/F. Whether the existing split points are adjusted after including a new binary rule
maxnumcut	the maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

**Details**

cox.main sequentially estimates a sequence of adaptive index models with up to "nsteps" terms for survival outcomes. The appropriate number of binary rules can be selected via K-fold cross-validation (cv.cox.main).

**Value**

cox.main returns maxsc, which is the partial likelihood score test statistics in the fitted model and res, which is a list with components

jmaa	number of predictors
cutp	split points for the binary rules
maxdir	direction of split: 1 represents "(x>cut)" and -1 represents "(x<cut)"
maxsc	observed partial likelihood score test statistics for the main effect

**Author(s)**

Lu Tian and Robert Tibshirani

**References**

Lu Tian and Robert Tibshirani (2010) "Adaptive index models for marker-based risk stratification", Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

**Examples**

```

## generate data
set.seed(1)

n=200
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
fail.time=rexp(n)*exp(-beta*z)
cen.time=rexp(n)*1.25
y=pmin(fail.time, cen.time)
y=round(y*10)/10
delta=1*(fail.time<cen.time)

## fit the main effect Cox AIM model
a=cox.main(x, y, delta, nsteps=10)

## examine the model sequence
print(a)

## compute the index based on the 2nd model of the sequence using data x
z.prd=index.prediction(a$res[[2]],x)

## compute the index based on the 2nd model of the sequence using new data xx, and compare the result with the true index
nn=10
xx=matrix(rnorm(nn*p), nn, p)
zz=(xx[,1]<0.2)+(xx[,5]>0.2)
zz.prd=index.prediction(a$res[[2]],xx)
cbind(zz, zz.prd)

```

---

cv.cox.interaction      *Cross-validation in the interaction Cox AIM*

---

**Description**

Cross-validation for selecting the number of binary rules in interaction AIM with survival outcomes in the context of Cox regression.

**Usage**

```
cv.cox.interaction(x, trt, y, status, K.cv=5, num.replicate=1, nsteps, mincut=0.1, backfit=F, maxnumcu
```

**Arguments**

x	n by p matrix. The covariate matrix
trt	n vector. The treatment indicator

<code>y</code>	<code>n</code> vector. The observed follow-up time
<code>status</code>	<code>n</code> 0/1 vector. The status indicator. 1=failure and 0=alive.
<code>K.cv</code>	K.cv-fold cross validation
<code>num.replicate</code>	number of independent replications of K-fold cross validations.
<code>nsteps</code>	the maximum number of binary rules to be included in the index
<code>backfit</code>	T/F. Whether the existing split points are adjusted after including a new binary rule
<code>mincut</code>	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
<code>maxnumcut</code>	the maximum number of binary splits per predictor
<code>dirp</code>	<code>p</code> vector. The given direction of the binary split for each of the <code>p</code> predictors. 0 represents "no pre-given direction"; 1 represents "( $x > \text{cut}$ )"; -1 represents "( $x < \text{cut}$ )". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

### Details

`cv.cox.interaction` implements K-fold cross-validation for the interaction Cox AIM. It estimates the partial likelihood score test statistics for testing the treatment\*index interaction in the test set. It also provides pre-validated fits for each observation and pre-validated partial likelihood score test statistics. The output can be used to select the optimal number of binary rules.

### Value

`cv.cox.interaction` returns

<code>kmax</code>	the optimal number of binary rules based the cross-validation
<code>meanscore</code>	<code>nsteps</code> -vector. The cross-validated partial likelihood score test statistics (significant at 0.05, if greater than 1.96) for the treatment*index interaction.
<code>pvfit.score</code>	<code>nsteps</code> -vector. The pre-validated partial likelihood score test statistics (significant at 0.05, if greater than 1.96) for the treatment*index interaction.
<code>preval</code>	<code>nsteps</code> by <code>n</code> matrix. Pre-validated fits for individual observation

### References

L Tian and R Tibshirani Adaptive index models for marker-based risk stratification, Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

R Tibshirani and B Efron, Pre-validation and inference in microarrays, *Statist. Appl. Genet. Mol. Biol.*, 1:1-18, 2002.

### Author(s)

Lu Tian and Robert Tibshirani

**Examples**

```

## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
trt=rbinom(n,1,0.5)
fail.time=rexp(n)*exp(-beta*z*trt)
cen.time=rexp(n)*1.25
y=pmin(fail.time, cen.time)
y=round(y*10)/10
delta=1*(fail.time<cen.time)

## cross-validate the interaction Cox AIM model
a=cv.cox.interaction(x, trt, y, delta, nsteps=10, K.cv=4, num.replicate=5)

## examine the score test statistics for the interaction in the test set
par(mfrow=c(1,2))
plot(a$meanscore, type="l")
plot(a$pvfit.score, type="l")

## construct the index with the optimal number of binary rules
k.opt=a$kmax
a=cox.interaction(x, trt, y, delta, nsteps=k.opt)
print(a)

```

---

cv.cox.main

*Cross-validation in main effect Cox AIM*


---

**Description**

Cross-validation for selecting the number of binary rules in the main effect AIM with survival outcomes.

**Usage**

```
cv.cox.main(x, y, status, K.cv=5, num.replicate=1, nsteps, mincut=0.1, backfit=F, maxnumcut=1, dirp=0)
```

**Arguments**

x	n by p matrix. The covariate matrix
y	n vector. The observed follow-up time
status	n 0/1 vector. The status indicator. 1=failure and 0=alive.

K.cv	K.cv-fold cross validation
num.replicate	number of independent replications of K-fold cross validations.
nsteps	the maximum number of binary rules to be included in the index
backfit	T/F. Whether the existing split points are adjusted after including new binary rules
mincut	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
maxnumcut	the maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

### Details

cv.cox.main implements the K-fold cross-validation for the main effect Cox AIM. It estimates the partial likelihood score test statistics in the test set for testing the association between the survival time and index constructed using training data. It also provides pre-validated fits for each observation and pre-validated partial likelihood score test statistics. The output can be used to select the optimal number of binary rules.

### Value

cv.cox.main returns

kmax	the optimal number of binary rules based the cross-validation
meanscore	nsteps-vector. The cross-validated partial likelihood score test statistics (significant at 0.05, if greater than 1.96) for the association between survival time and index.
pvfit.score	nsteps-vector. The pre-validated partial likelihood score test statistics (significant at 0.05, if greater than 1.96) for the association between survival time and index.
preval	nsteps by n matrix. Pre-validated fits for individual observation

### References

L Tian and R Tibshirani Adaptive index models for marker-based risk stratification, Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

R Tibshirani and B Efron, Pre-validation and inference in microarrays, Statist. Appl. Genet. Mol. Biol., 1:1-18, 2002.

### Author(s)

Lu Tian and Robert Tibshirani

**Examples**

```

## generate data

set.seed(1)

n=200
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
fail.time=rexp(n)*exp(-beta*z)
cen.time=rexp(n)*1.25
y=pmin(fail.time, cen.time)
y=round(y*10)/10
delta=1*(fail.time<cen.time)

## cross-validate the main effect Cox AIM
a=cv.cox.main(x, y, delta, nsteps=10, K.cv=3, num.replicate=3)

## examine the test statistics in the test set
par(mfrow=c(1,2))
plot(a$meanscore, type="l")
plot(a$pvfit.score, type="l")

## construct the index with the optimal number of binary rules
k.opt=a$kmax
a=cox.main(x, y, delta, nsteps=k.opt)
print(a)

```

---

cv.lm.interaction      *Cross-validation in interaction linear AIM*

---

**Description**

Cross-validation for selecting the number of binary rules in the interaction AIM with continuous outcomes

**Usage**

```
cv.lm.interaction(x, trt, y, K.cv=5, num.replicate=1, nsteps, mincut=0.1, backfit=F, maxnumcut=1, dirp
```

**Arguments**

x	n by p matrix. The covariate matrix
trt	n vector. The treatment indicator

<code>y</code>	<code>n</code> vector. The continuous response variable
<code>K.cv</code>	<code>K</code> .cv-fold cross validation
<code>num.replicate</code>	number of independent replications of <code>K</code> -fold cross validations
<code>nsteps</code>	the maximum number of binary rules to be included in the index
<code>mincut</code>	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
<code>backfit</code>	T/F. Whether the existing split points are adjusted after including a new binary rule
<code>maxnumcut</code>	the maximum number of binary splits per predictor
<code>dirp</code>	<code>p</code> vector. The given direction of the binary split for each of the <code>p</code> predictors. 0 represents "no pre-given direction"; 1 represents "( $x > \text{cut}$ )"; -1 represents "( $x < \text{cut}$ )". Alternatively, " <code>dirp=0</code> " represents that there is no pre-given direction for any of the predictor.

### Details

`cv.lm.interaction` implements the `K`-fold cross-validation for interaction linear AIM. It estimates the score test statistics in the test set for testing the `treatment*index` interaction. It also provides the pre-validated fits for each observation and pre-validated score test statistics. The output can be used to select the optimal number of binary rules.

### Value

`cv.lm.interaction` returns

<code>kmax</code>	the optimal number of binary rules based the cross-validation
<code>meanscore</code>	<code>nsteps</code> -vector. The cross-validated score test statistics (significant at 0.05, if greater than 1.96) for the <code>treatment*index</code> interaction
<code>pvfit.score</code>	<code>nsteps</code> -vector. The pre-validated score test statistics (significant at 0.05, if greater than 1.96) for the <code>treatment*index</code> interaction.
<code>preval</code>	<code>nsteps</code> by <code>n</code> matrix. Prevalidated fits for individual observation

### References

L Tian and R Tibshirani Adaptive index models for marker-based risk stratification, Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

R Tibshirani and B Efron, Pre-validation and inference in microarrays, *Statist. Appl. Genet. Mol. Biol.*, 1:1-18, 2002.

### Author(s)

Lu Tian and Robert Tibshirani

**Examples**

```

## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
trt=rbinom(n, 1, 0.5)
beta=1
y=trt+beta*trt*z+rnorm(n)

## cross-validate the interaction linear AIM
a=cv.lm.interaction(x, trt, y, nsteps=10, K.cv=5, num.replicate=3)

## examine the score test statistics in the test set
par(mfrow=c(1,2))
plot(a$meanscore, type="l")
plot(a$pvfit.score, type="l")

## construct the index with the optimal number of binary rules
k.opt=a$kmax
a=lm.interaction(x, y, trt, nsteps=k.opt)
print(a)

```

---

cv.lm.main

*Cross-validation in main effect linear AIM*


---

**Description**

Cross-validation for selecting the number of binary rules in the main effect linear AIM

**Usage**

```
cv.lm.main(x, y, K.cv=5, num.replicate=1, nsteps, mincut=0.1, backfit=F, maxnumcut=1, dirp=0)
```

**Arguments**

x	n by p matrix. The covariate matrix
y	n vector. The continuous response variable
K.cv	K.cv-fold cross validation
num.replicate	number of independent replications of K-fold cross validations.
nsteps	the maximum number of binary rules to be included in the index
mincut	the minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.

backfit	T/F. Whether the existing split points are adjusted after including new a binary rule
maxnumcut	the maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

### Details

cv.lm.main implements the K-fold cross-validation for the main effect linear AIM. It estimates the score test statistics in the test set for testing the association between the continuous response and index constructed using training data. It also provides pre-validated fits for each observation and the pre-validated score test statistics. The output can be used to select the optimal number of binary rules.

### Value

cv.lm.main	returns
kmax	the optimal number of binary rules based the cross-validation
meanscore	nsteps-vector. The cross-validated score test statistics (significant at 0.05, if greater than 1.96) for the association between survival time and index.
pvfit.score	nsteps-vector. The pre-validated score test statistics (significant at 0.05, if greater than 1.96) for the association between survival time and index.
preval	nsteps by n matrix. Pre-validated fits for individual observation

### References

L Tian and R Tibshirani Adaptive index models for marker-based risk stratification, Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

R Tibshirani and B Efron, Pre-validation and inference in microarrays, Statist. Appl. Genet. Mol. Biol., 1:1-18, 2002.

### Author(s)

Lu Tian and Robert Tibshirani

### Examples

```
## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
y=beta*z+rnorm(n)
```

```
## cross-validate the linear main effects AIM
a=cv.lm.main(x, y, nsteps=10, K.cv=5, num.replicate=3)

## examine score test statistics in the test set
par(mfrow=c(1,2))
plot(a$meanscore, type="l")
plot(a$pvfit.score, type="l")

## construct the index with the optimal number of binary rules
k.opt=a$kmax
a=lm.main(x, y, nsteps=k.opt)
print(a)
```

---

cv.logistic.interaction

*Cross-validation in interaction logistic AIM*

---

## Description

Cross-validation for selecting the number of binary rules in the interaction AIM with binary outcomes.

## Usage

```
cv.logistic.interaction(x, trt, y, K.cv=5, num.replicate=1, nsteps, mincut=0.1, backfit=F, maxnumcut=1)
```

## Arguments

x	n by p matrix. The covariate matrix
trt	n vector. The treatment indicator
y	n 0/1 vector. The binary response variable
K.cv	K.cv-fold cross validation
num.replicate	number of independent replications of K-fold cross validations.
nsteps	the maximum number of binary rules to be included in the index
mincut	The minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
backfit	T/F. Whether the existing split points are adjusted after including a new binary rule
maxnumcut	The maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.
weight	a positive value. The weight given to responses: "weight=0" means that all observations are equally weighted.

**Details**

`cv.logistic.interaction` implements the K-fold cross-validation for the interaction logistic AIM. It estimates the score test statistics in the test set for testing the treatment\*index interaction. It also provides pre-validated fits for each observation and pre-validated score test statistic. The output can be used to select the optimal number of binary rules.

**Value**

`cv.logistic.interaction` returns

<code>kmax</code>	the optimal number of binary rules based the cross-validation
<code>meanscore</code>	nsteps-vector. The cross-validated score test statistics (significant at 0.05, if greater than 1.96) for the treatment*index interaction.
<code>pvfit.score</code>	nsteps-vector, the pre-validated score test statistics (significant at 0.05, if greater than 1.96) for the treatment*index interaction.
<code>preval</code>	nsteps by n matrix. Prevalidated fits for individual observation

**References**

L Tian and R Tibshirani Adaptive index models for marker-based risk stratification, Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

R Tibshirani and B Efron, Pre-validation and inference in microarrays, *Statist. Appl. Genet. Mol. Biol.*, 1:1-18, 2002.

**Author(s)**

Lu Tian and Robert Tibshirani

**Examples**

```
## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
trt=rbinom(n,1, 0.5)
beta=1
prb=1/(1+exp(trt-beta*trt*z-0.5))
y=rbinom(n,1,prb)

## cross-validate the logistic interaction AIM
a=cv.logistic.interaction(x, trt, y, nsteps=10, K.cv=4, num.replicate=5)

## examine score test statistics in the test set
par(mfrow=c(1,2))
plot(a$meanscore, type="l")
plot(a$pvfit.score, type="l")
```

```
## construct the index with the optimal number of binary rules
k.opt=a$kmax
a=logistic.interaction(x, trt, y, nsteps=k.opt)
print(a)
```

---

cv.logistic.main      *Cross-validation in the main effect logistic AIM*

---

### Description

Cross-validation for selecting the number of binary rules in the main effect AIM with binary outcomes

### Usage

```
cv.logistic.main(x, y, K.cv=5, num.replicate=1, nsteps, mincut=0.1, backfit=F, maxnumcut=1, dirp=0, weight=1)
```

### Arguments

x	n by p matrix. The covariate matrix
y	n 0/1 vector. The binary response variable
K.cv	K.cv-fold cross validation
num.replicate	number of independent replications of K-fold cross validations.
nsteps	The maximum number of binary rules to be included in the index
mincut	The minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
backfit	T/F. Whether the existing split points are adjusted after including a new binary rule
maxnumcut	The maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.
weight	a positive value. The weight given to responses. "weight=0" means that all observations are equally weighted.

### Details

cv.logistic.main implements the K-fold cross-validation for the main effect logistic AIM. It estimates the score test statistics in the test set for testing the association between the binary outcome and index constructed using training data. It also provides pre-validated fits for each observation and the pre-validated score test statistic. The output can be used to select the optimal number of binary rules.

**Value**

cv.lm.main returns

kmax	the optimal number of binary rules based the cross-validation
meanscore	nsteps-vector. The cross-validated score test statistics (significant at 0.05, if greater than 1.96) for the association between survival time and index.
pvfit.score	nsteps-vector. The pre-validated score test statistics (significant at 0.05, if greater than 1.96) for the association between survival time and index.
preval	nsteps by n matrix. Pre-validated fits for individual observation

**References**

L Tian and R Tibshirani Adaptive index models for marker-based risk stratification, Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

R Tibshirani and B Efron, Pre-validation and inference in microarrays, Statist. Appl. Genet. Mol. Biol., 1:1-18, 2002.

**Author(s)**

Lu Tian and Robert Tibshirani

**Examples**

```
## generate data
set.seed(1)
n=500
p=20

x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
prb=1/(1+exp(-beta*z))
y=rbinom(n,1,prb)

## cross-validate the logistic main effects AIM
a=cv.logistic.main(x, y, nsteps=10, K.cv=5, num.replicate=3)

## examine the score test statistics in the test set
par(mfrow=c(1,2))
plot(a$meanscore, type="l")
plot(a$pvfit.score, type="l")

## construct the index with the optimal number of binary rules
k.opt=a$kmax
a=logistic.main(x, y, nsteps=k.opt)
print(a)
```

---

index.prediction	<i>Predict index based on fitted AIM</i>
------------------	--

---

**Description**

Compute the index for new observations using output from `lm.main`, `lm.interaction`, `logistic.main`, `logistic.interaction`, `cox.main` and `cox.interaction`.

**Usage**

```
index.prediction(res, x)
```

**Arguments**

<code>res</code>	list "res" term from the outputs in <code>lm.main</code> , <code>lm.interaction</code> , <code>logistic.main</code> , <code>logistic.interaction</code> , <code>cox.main</code> and <code>cox.interaction</code>
<code>x</code>	New covariate matrix

**Details**

`index.prediction` computes the new index for given observations based on the fitted AIM

**Value**

`index.prediction` returns score which is the index for new observations with covariate matrix "x".

**Author(s)**

Lu Tian and Robert Tibshirani

**References**

Lu Tian and Robert Tibshirani (2010) Adaptive index models for marker-based risk stratification. Tech Report. Available at <http://www-stat.stanford.edu/~tibs/AIM>.

**Examples**

```
## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
trt=rbinom(n,1, 0.5)
beta=1
prb=1/(1+exp(trt-beta*trt*z-0.5))
y=rbinom(n,1,prb)
```

```

## fit the interaction logistic AIM model
a=logistic.interaction(x, trt, y, nsteps=10)

## examine the model sequence
print(a)

## compute the index based on the 2nd model of the sequence, using data x
z.prd=index.prediction(a$res[[2]],x)

## compute the index based on the 2nd model of the sequence using new data xx, and compare the result with the true index
nn=10
xx=matrix(rnorm(nn*p), nn, p)
zz=(xx[,1]<0.2)+(xx[,5]>0.2)
zz.prd=index.prediction(a$res[[2]],xx)
cbind(zz, zz.prd)

```

---

<code>lm.interaction</code>	<i>Interaction linear adaptive index model</i>
-----------------------------	--

---

### Description

Estimate adaptive index model for continuous outcomes in the context of linear regression. The resulting index characterizes the interactions between the covariates and treatment.

### Usage

```
lm.interaction(x, trt, y, nsteps=8, backfit=F, mincut=.1, maxnumcut=1, dirp=0)
```

### Arguments

<code>x</code>	n by p matrix. The covariate matrix
<code>trt</code>	n vector. The treatment indicator
<code>y</code>	n vector. The continuous response variable
<code>nsteps</code>	the maximum number of binary rules to be included in the index
<code>backfit</code>	T/F. Whether the existing split points are adjusted after including a new binary rule
<code>mincut</code>	The minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
<code>maxnumcut</code>	The maximum number of binary splits per predictor
<code>dirp</code>	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

**Details**

lm.interaction sequentially estimates a sequence of adaptive index models with up to "nsteps" terms for continuous outcomes. The algorithm seeks the index having the strong interaction with the treatment in the linear model. The appropriate number of binary rules can be selected via K-fold cross-validation (cv.lm.interaction).

**Value**

lm.interaction returns maxsc, which is the observed score test statistics for the index\*treatment interaction in the fitted model and res, which is a list with components

jmaa	number of predictors
cutp	split points for the binary rules
maxdir	direction of split: 1 represents "(x>cut)" and -1 represents "(x<cut)"
maxsc	observed score test statistics for the interaction

**Author(s)**

Lu Tian and Robert Tibshirani

**References**

Lu Tian and Robert Tibshirani (2010) "Adaptive index models for marker-based risk stratification", Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

**Examples**

```
## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
trt=rbinom(n, 1, 0.5)
beta=1
y=trt+beta*trt*z+rnorm(n)

## fit the interaction linear AIM
a=lm.interaction(x, trt, y, nsteps=10)

## examine the model sequence
print(a)

## compute the index based on the 2nd model of the sequence, using data x
z.prd=index.prediction(a$res[[2]],x)

## compute the index based on the 2nd model of the sequence using new data xx, and compare the result with the true index
nn=10
```

```
xx=matrix(rnorm(nn*p), nn, p)
zz=(xx[,1]<0.2)+(xx[,5]>0.2)
zz.prd=index.prediction(a$res[[2]],xx)
cbind(zz, zz.prd)
```

---

lm.main *Main effect linear adaptive index model*

---

### Description

Estimate adaptive index model for continuous outcomes in the context of linear regression. The resulting index characterizes the main covariate effect on the continuous response.

### Usage

```
lm.main(x, y, nsteps=8, backfit=F, mincut=.1, maxnumcut=1, dirp=0)
```

### Arguments

x	n by p matrix. The covariate matrix
y	n vector. The continuous response variable
nsteps	the maximum number of binary rules to be included in the index
backfit	T/F. Whether the existing split points are adjusted after including a new binary rule
mincut	The minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
maxnumcut	The maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.

### Details

lm.main sequentially estimates a sequence of adaptive index models with up to "nsteps" terms for continuous outcomes. The appropriate number of binary rules can be selected via K-fold cross-validation(cv.lm.main).

### Value

lm.main returns maxsc, which is the score test statistics achieved in the fitted model and res, which is a list with components

jmaa	number of predictors
cutp	split points for the binary rules
maxdir	direction of split: 1 represents "(x>cut)" and -1 represents "(x<cut)"
maxsc	observed score test statistics for the main effect

**Author(s)**

Lu Tian and Rob Tibshirani

**References**

Lu Tian and Robert Tibshirani (2010) "Adaptive index models for marker-based risk stratification", Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

**Examples**

```
## generate data
set.seed(1)

n=500
p=20
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
y=beta*z+rnorm(n)

## fit the main effects linear AIM
a=lm.main(x, y, nsteps=10)

## examine the model sequence
print(a)

## compute the index based on the 2nd model of the sequence using data x
z.prd=index.prediction(a$res[[2]],x)

## compute the index based on the 2nd model of the sequence using new data xx, and compare the result with the true index
nn=10
xx=matrix(rnorm(nn*p), nn, p)
zz=(xx[,1]<0.2)+(xx[,5]>0.2)
zz.prd=index.prediction(a$res[[2]],xx)
cbind(zz, zz.prd)
```

---

logistic.interaction    *Interaction logistic adaptive index model*

---

**Description**

Estimate adaptive index model for binary outcomes in the context of logistic regression. The resulting index characterizes the interaction between the covariates and treatment.

**Usage**

```
logistic.interaction(x, trt, y, nsteps=8, mincut=.1, backfit=F, maxnumcut=1, dirp=0, weight=1)
```

**Arguments**

<code>x</code>	n by p matrix. The covariate matrix
<code>trt</code>	n vector. The treatment indicator
<code>y</code>	n 0/1 vector. The binary response variable
<code>nsteps</code>	the maximum number of binary rules to be included in the index
<code>mincut</code>	The minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
<code>backfit</code>	T/F. Whether the existing split points are adjusted after including a new binary rule
<code>maxnumcut</code>	The maximum number of binary splits per predictor
<code>dirp</code>	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.
<code>weight</code>	a positive value. The weight given to responses. "weight=0" means that all observations are equally weighted.

**Details**

`logistic.interaction` sequentially estimates a sequence of adaptive index models with up to "nsteps" terms for binary outcomes. The algorithm seeks the index having the strong interaction with the treatment in the logistic regression model. The appropriate number of binary rules can be selected via K-fold cross-validation (`cv.logistic.interaction`).

**Value**

`logistic.interaction` returns `maxsc`, which is the observed score test statistics for the index\*treatment interaction in the fitted model and `res`, which is a list with components

<code>jmaa</code>	number of predictors
<code>cutp</code>	split points for the binary rules
<code>maxdir</code>	direction of split: 1 represents "(x>cut)" and -1 represents "(x<cut)"
<code>maxsc</code>	observed score test statistics for the interaction

**Author(s)**

Lu Tian and Robert Tibshirani

**References**

Lu Tian and Robert Tibshirani (2010) "Adaptive index models for marker-based risk stratification", Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

**Examples**

```

## generate data
set.seed(1)

n=400
p=10
x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
trt=rbinom(n,1, 0.5)
beta=1
prb=1/(1+exp(trt-beta*trt*z-0.5))
y=rbinom(n,1,prb)

## fit the interaction logistic AIM
a=logistic.interaction(x, trt, y, nsteps=10)

## examine the model sequence
print(a)

## compute the index based on the 2nd model of the sequence using data x
z.prd=index.prediction(a$res[[2]],x)

## compute the index based on the 2nd model of the sequence using new data xx, and compare the result with the true index
nn=10
xx=matrix(rnorm(nn*p), nn, p)
zz=(xx[,1]<0.2)+(xx[,5]>0.2)
zz.prd=index.prediction(a$res[[2]],xx)
cbind(zz, zz.prd)

```

---

logistic.main

*Main effect logistic adaptive index model*


---

**Description**

Estimate adaptive index model for binary outcomes in the context of logistic regression. The resulting index characterizes the main covariate effect on the response probability.

**Usage**

```
logistic.main(x, y, nsteps=8, mincut=.1, backfit=F, maxnumcut=1, dirp=0, weight=1)
```

**Arguments**

x	n by p matrix. The covariate matrix
y	n 0/1 vector. The binary response variable
nsteps	the maximum number of binary rules to be included in the index
backfit	T/F. Whether the existing split points are adjusted after including a new binary rule

mincut	The minimum cutting proportion for the binary rule at either end. It typically is between 0 and 0.2.
maxnumcut	The maximum number of binary splits per predictor
dirp	p vector. The given direction of the binary split for each of the p predictors. 0 represents "no pre-given direction"; 1 represents "(x>cut)"; -1 represents "(x<cut)". Alternatively, "dirp=0" represents that there is no pre-given direction for any of the predictor.
weight	a positive number. The weight given to responses. "weight=0" means that all observations are equally weighted.

### Details

logistic.main sequentially estimates a sequence of adaptive index models with up to "nsteps" terms for binary outcomes. The appropriate number of binary rules can be selected via K-fold cross-validation(cv.logistic.main).

### Value

logistic.main returns maxsc, which is the score test statistics achieved in the fitted model and res, which is a list with components

jmaa	number of predictors
cutp	split points for the binary rules
maxdir	direction of split: 1 represents "(x>cut)" and -1 represents "(x<cut)"
maxsc	observed score test statistics for the main effect

### Author(s)

Lu Tian and Robert Tibshirani

### References

Lu Tian and Robert Tibshirani (2010) "Adaptive index models for marker-based risk stratification", Tech Report, available at <http://www-stat.stanford.edu/~tibs/AIM>.

### Examples

```
## generate data
set.seed(1)
n=200
p=10

x=matrix(rnorm(n*p), n, p)
z=(x[,1]<0.2)+(x[,5]>0.2)
beta=1
prb=1/(1+exp(-beta*z))
y=rbinom(n,1,prb)
```

```
## fit logistic main effects AIM
a=logistic.main(x, y, nsteps=10)

## examine the model sequence
print(a)

## compute the index based on the 2nd model of the sequence using data x
z.prd=index.prediction(a$res[[2]],x)

## compute the index based on the 2nd model of the sequence using new data xx, and compare the result with the true in
nn=10
xx=matrix(rnorm(nn*p), nn, p)
zz=(xx[,1]<0.2)+(xx[,5]>0.2)
zz.prd=index.prediction(a$res[[2]],xx)
cbind(zz, zz.prd)
```

# Index

backfit.cox.interaction, 2  
backfit.cox.main, 2  
backfit.lm.interaction, 2  
backfit.lm.main, 3  
backfit.logistic.interaction, 3  
backfit.logistic.main, 3

cox.interaction, 4  
cox.main, 5  
cv.cox.interaction, 7  
cv.cox.main, 9  
cv.lm.interaction, 11  
cv.lm.main, 13  
cv.logistic.interaction, 15  
cv.logistic.main, 17

index.prediction, 19

lm.interaction, 20  
lm.main, 22  
logistic.interaction, 23  
logistic.main, 25