

# Package: ROCSI (via r-universe)

September 5, 2024

**Type** Package

**Title** Receiver Operating Characteristic Based Signature Identification

**Version** 0.1.0

**Description** Optimal linear combination predictive signatures for maximizing the area between two Receiver Operating Characteristic (ROC) curves (treatment vs. control).

**License** GPL (>= 2)

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Imports** glmnet, MASS

**NeedsCompilation** no

**Author** Xin Huang [aut, cre, cph]

**Maintainer** Xin Huang <xin.huang@abbvie.com>

**Repository** CRAN

**Date/Publication** 2022-08-24 09:12:36 UTC

## Contents

AUC . . . . .	2
beta2theta . . . . .	2
C.index . . . . .	3
cvfolds0 . . . . .	4
data.gen . . . . .	4
grad.sub . . . . .	6
gradsqr . . . . .	6
hessAUC . . . . .	7
hessAUC.sub . . . . .	8
HIC . . . . .	8
MClogit . . . . .	9
pair.diff . . . . .	11
pair.diff.surv . . . . .	11
ROCSI . . . . .	12
theta2beta . . . . .	14

**Index** **15**

AUC *AUC*

### Description

Empirical AUC estimate

### Usage

AUC(outcome, predict)

### Arguments

outcome	binary outcome (1: desired outcome; 0: otherwise)
predict	prediction score

### Details

Function for AUC when input is X and Y.

### Value

a numeric value of empirical estimation of area under the ROC curves

### Examples

```
# no run
```

beta2theta *beta2theta*

### Description

Function to translate beta into theta, the n-sphere constrain

### Usage

beta2theta(beta)

### Arguments

beta	estimates of coefficient beta
------	-------------------------------

### Details

Function to translate beta into theta, the n-sphere constrain

**Value**

a numeric vector for theta (dimension-1)

**Examples**

```
# no run
```

---

<code>C.index</code>	<i>C.index</i>
----------------------	----------------

---

**Description**

Empirical c-index estimate

**Usage**

```
C.index(yvar, score, censorvar, data)
```

**Arguments**

yvar	column name for observed time
score	column name for marker value
censorvar	column name for censor (1 is event, 0 is censored)
data	input data matrix

**Details**

Function for c-index when input is X and Y.

**Value**

a numeric value of empirical estimation of c-index

**Examples**

```
# no run
```

*cvfolds0**cvfolds0*

---

**Description**

internal function for generating CV fold index

**Usage**

```
cvfolds0(X, Y, idx, nfolds = 5)
```

**Arguments**

X	marker matrix for non-responders
Y	marker matrix for responders
idx	m*n by 2 matrix for row index of marker matrix, first column is row index in X; second column is for Y
nfolds	the cross-validation folds

**Details**

Function for generate CV fold index

**Value**

a vector containing CV fold index for each row in Z

**Examples**

```
# no run
```

---

*data.gen**data.gen*

---

**Description**

Function for simulated data generation

**Usage**

```
data.gen(
  n,
  k,
  prevalence = sqrt(0.5),
  prog.eff = 1,
  sig2,
  y.sig2,
  rho,
  rhos.bt.real,
  a.constant
)
```

**Arguments**

n	Total sample size
k	Number of markers
prevalence	prevalence of predictive biomarkers with values above the cutoff
prog.eff	effect size <i>beta</i> for prognostic biomarker
sig2	standard deviation of each marker
y.sig2	Standard Deviation of the error term in the linear component
rho	$\rho \cdot \text{sig}^2$ is the entries for covariance matrix between pairs of different k markers
rhos.bt.real	correlation between each prognostic and predictive markers
a.constant	a constant is set such that there is no overall treatment effect

**Details**

Function for simulated data generation

**Value**

A list of simulated clinical trial data with heterogeneous prognostic and predictive biomarkers

**Examples**

```
n <- 500
k <- 10
prevalence <- sqrt(0.5)
rho <- 0.2
sig2 <- 2
rhos.bt.real <- c(0, rep(0.1, (k-3))) * sig2
y.sig2 <- 1
prog.eff <- 0.5
effect.size <- 1
a.constant <- effect.size / (2 * (1 - prevalence))
ObsData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
```

---

grad.sub	<i>grad.sub</i>
----------	-----------------

---

**Description**

Internal function of grad\_square in the GCV

**Usage**

```
grad.sub(z, beta)
```

**Arguments**

z	(m x n) x p data matrix as prepared for ROCSI
beta	estimates of coefficient beta

**Details**

Internal function of grad\_square in the GCV

**Value**

grad\_square in the GCV

**Examples**

```
# no run
```

---

gradsqr	<i>gradsqr</i>
---------	----------------

---

**Description**

Internal function for HIC calculation

**Usage**

```
gradsqr(beta, Z0, index, w = 1)
```

**Arguments**

beta	estimates of coefficient beta
Z0	(m x n) x p Z matrix as prepared for ROCSI
index	m*n by 2 matrix for the subindex for the pair difference in Z
w	a vector of weights Z (can be used for inverse probability weighting for missing data, default is 1)

**Details**

Internal function for HIC calculation

**Value**

gradient square for the GCV.

**Examples**

```
# no run
```

---

<code>hessAUC</code>	<i>hessAUC</i>
----------------------	----------------

---

**Description**

function for Hessian matrix of AUC

**Usage**

```
hessAUC(beta, Z, w = 1)
```

**Arguments**

- beta estimates of coefficient beta
- Z (m x n) x p data matrix as prepared for ROCSI
- w a vector of weights Z (can be used for inverse probability weighting for missing data, default is 1)

**Details**

function for Hessian matrix of AUC

**Value**

Hessian matrix of AUC.

**Examples**

```
# no run
```

---

 hessAUC.sub

*hessAUC.sub*


---

**Description**

Internal function for hessAUC

**Usage**

hessAUC.sub(z, beta)

**Arguments**

z (m x n) x p data matrix as prepared for ROCSI  
 beta estimates of coefficient beta

**Details**

Internal function for hessAUC

**Value**

Hessian matrix components.

**Examples**

```
# no run
```

---

 HIC

*HIC*


---

**Description**

function for HIC calculation

**Usage**

HIC(beta, Z, index, w = 1)

**Arguments**

beta estimates of coefficient beta  
 Z matrix prepared for ROCSI  
 index m\*n by 2 matrix for the subindex for the pair difference in Z  
 w a vector of weights Z (can be used for inverse probability weighting for missing data, default is 1)



**Details**

Function for HIC calculation

**Value**

A numeric value with corresponding HIC

**Examples**

```
# no run
```

---

MClogit	<i>MClogit</i>
---------	----------------

---

**Description**

function for modified covariate methods based on glmnet

**Usage**

```
MClogit(
  dataset,
  yvar,
  xvars,
  trtvar,
  cvar = NULL,
  nfolds = 5,
  type = "binary",
  newx = NULL,
  bestsub = "lambda.1se",
  type.measure = "auc"
)
```

**Arguments**

dataset	data matrix for training dataset
yvar	column name for outcome
xvars	a string vector of column names for input markers
trtvar	column name for treatment (the column should contain binary code with 1 being treatment and 0 being control)
cvar	column name for censor (the column should contain binary code with 1 being event and 0 being censored)
nfolds	n fold CV used for cv.glmnet
type	outcome type ("binary" for binary outcome and "survival" for time-to-event outcome)

newx            data matrix for testing dataset X  
 bestsub        criteria for best lambda, used by glmnet  
 type.measure   type of measure used by glmnet

## Details

function for ROCSI

## Value

A list with ROCSI output

**x.logit** final beta estimated from MClogit

**predScore** a data.frame of testing data and its predictive signature scores (based on beta.aABC) for each subjects

**abc** ABC in testing dataset based on optimal beta

**fit.cv** the fitted glmnet object

## Examples

```
n <- 100
k <- 5
prevalence <- sqrt(0.5)
rho<-0.2
sig2 <- 2
rhos.bt.real <- c(0, rep(0.1, (k-3)))*sig2
y.sig2 <- 1
yvar="y.binary"
xvars=paste("x", c(1:k), sep="")
trtvar="treatment"
prog.eff <- 0.5
effect.size <- 1
a.constant <- effect.size/(2*(1-prevalence))
ObsData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
TestData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
bst.mod <- MClogit(dataset=ObsData$data, yvar=yvar, xvars=xvars,
  trtvar=trtvar, nfolds = 5, newx=TestData$data,
  type="binary", bestsub="lambda.1se")
bst.mod$abc
bst.mod$x.logit[-1,1]
```

---

pair.diff                      *pair.diff*

---

**Description**

internal function for generating Z matrix (binary endpoint)

**Usage**

```
pair.diff(X, Y, A)
```

**Arguments**

X	marker matrix for non-responders
Y	marker matrix for responders
A	Treatment arm indicator (1 is treatment, 0 is control)

**Details**

Function for generate Z matrix for binary endpoint

**Value**

A list of prepared data input for ROCSI

**Examples**

```
# no run
```

---

pair.diff.surv                      *pair.diff.surv*

---

**Description**

internal function for generating Z matrix (time-to-event endpoint)

**Usage**

```
pair.diff.surv(X, Y, A, C)
```

**Arguments**

X	marker matrix
Y	a vector for observed time
A	a vector for Treatment arm indicator (1 is treatment, 0 is control)
C	a vector for censor (1 is event, 0 is censored)

**Details**

Function for generate Z matrix for time-to-event endpoint

**Value**

A list of prepared data input for ROCSI

**Examples**

```
# no run
```

---

ROCSI

*ROCSI*

---

**Description**

function for ROCSI

**Usage**

```
ROCSI(
  Dtrain,
  Dtest = NULL,
  yvar,
  xvars,
  trtvar,
  cvar = NULL,
  n folds = 5,
  type = "binary"
)
```

**Arguments**

Dtrain	data matrix for training dataset
Dtest	optional data matrix for testing dataset
yvar	column name for outcome
xvars	a string vector of column names for input markers
trtvar	column name for treatment (the column should contain binary code with 1 being treatment and 0 being control)
cvar	column name for censor (the column should contain binary code with 1 being event and 0 being censored)
n folds	n fold CV used for cv.glmnet
type	outcome type ("binary" for binary outcome and "survival" for time-to-event outcome)

**Details**

function for ROCSI

**Value**

A list with ROCSI output

**beta.aABC** final beta estimated from ROCSI based on  $ABC^{(acv)}$

**beta.1se** final beta estimated from lambda.1se based on nfold CV

**lambda.aABC** optimal lambda selected by optimizing  $ABC^{(acv)}$

**fit.cv** fitted cv.glmnet model

**log** log matrix of all lambdas and ABCs

**abc.test** ABC in testing dataset based on optimal beta

**abc.test1se** ABC in testing dataset based on 1se beta

**predScore** a data.frame of testing data and its predictive signature scores (based on beta.aABC) for each subjects

**predScore.1se** a data.frame of testing data and its predictive signature scores (based on beta.1se) for each subjects

**Examples**

```
n <- 100
k <- 5
prevalence <- sqrt(0.5)
rho<-0.2
sig2 <- 2
rhos.bt.real <- c(0, rep(0.1, (k-3)))*sig2
y.sig2 <- 1
yvar="y.binary"
xvars=paste("x", c(1:k), sep="")
trtvar="treatment"
prog.eff <- 0.5
effect.size <- 1
a.constant <- effect.size/(2*(1-prevalence))
ObsData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
TestData <- data.gen(n=n, k=k, prevalence=prevalence, prog.eff=prog.eff,
  sig2=sig2, y.sig2=y.sig2, rho=rho,
  rhos.bt.real=rhos.bt.real, a.constant=a.constant)
bst.aabc <- ROCSI(Dtrain=ObsData$data, Dtest = TestData$data, yvar=yvar,
  xvars=xvars, trtvar=trtvar, cvar=NULL, nfolds=5, type="binary")
bst.aabc$beta.aABC
bst.aabc$log
bst.aabc$abc.test
bst.aabc$beta.1se
bst.aabc$abc.test1se
```

---

theta2beta	<i>theta2beta</i>
------------	-------------------

---

**Description**

Function to translate theta into beta

**Usage**

```
theta2beta(theta)
```

**Arguments**

theta            n-sphere coordination

**Details**

Function to translate beta into theta, the n-sphere constrain

**Value**

a numeric vector for beta (dimension+1)

**Examples**

```
# no run
```

# Index

AUC, [2](#)

beta2theta, [2](#)

C.index, [3](#)  
cvfolds0, [4](#)

data.gen, [4](#)

grad.sub, [6](#)  
gradsqr, [6](#)

hessAUC, [7](#)  
hessAUC.sub, [8](#)  
HIC, [8](#)

MClogit, [9](#)

pair.diff, [11](#)  
pair.diff.surv, [11](#)

ROCSI, [12](#)

theta2beta, [14](#)