Package 'DTRlearn2'

January 20, 2025

Title Statistical Learning Methods for Optimizing Dynamic Treatment
Regimes

Version 1.1

Author Yuan Chen, Ying Liu, Donglin Zeng, Yuanjia Wang

Maintainer Yuan Chen <irene.yuan.chen@gmail.com>

Description

We provide a comprehensive software to estimate general K-stage DTRs from SMARTs with Q-
learning and a variety of outcome-weighted learning methods. Penalizations are allowed for vari-
able selection and model regularization. With the outcome-weighted learning scheme, differ-
ent loss functions - SVM hinge loss, SVM ramp loss, binomial deviance loss, and L2 loss -
are adopted to solve the weighted classification problem at each stage; augmentation in the out-
comes is allowed to improve efficiency. The estimated DTR can be easily applied to a new sam-
ple for individualized treatment recommendations or DTR evaluation.

Depends kernlab,MASS,Matrix,foreach,glmnet, R (>= 2.10)

License GPL-2

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-22 15:52:05 UTC

Contents

---

adhd   *A 2-stage SMART data of children with ADHD*

Description

We provide a two-stage sequential multiple assignment randomized trial (SMART) data of 150 children with ADHD mimicking a real world study. At the first stage, children were randomized to treatment of low-intensity behavioral modification (BMOD) or low-intensity methamphetamine (MED) with equal probability. At second stage, children were randomized to treatment of low-intensity BMOD + low-intensity MED, or high-intensity BMOD with equal probability. The primary outcome of study was children's school performance score ranging from 1 to 5 assessed at the end of the study for all participants.

Usage

```
data("adhd")
```

Format

A data frame with 150 observations on the following 11 variables.

id IDs of the 150 children

o11 baseline covariate coded as 0/1: diagnosed with ODD (oppositional defiant disorder) before the first-stage intervention

o12 baseline covariate: ADHD score at the end of the previous school year (ranging from 0 to 3, larger values for fewer ADHD symptoms)

o13 baseline covariate coded as 0/1: receiving medication during the previous school year

o14 baseline covariate coded as 0/1: race - white (coded 1) versus nonwhite (coded 0)

a1 first-stage intervention coded as -1/1: -1 for low-intensity methamphetamine (MEDS), 1 for low-intensity behavioral modification (BMOD)

r first-stage response indicator coded as 0/1

o21 intermediate outcome: number of months until non-response (maximum: 8 months, NA for responders)

o22 intermediate outcome coded as 0/1: adherence to the first-stage intervention, 1 for high adherence

a2 second-stage intervention coded as -1/1: -1 for low-intensity BMOD + MEDS, 1 for high-intensity BMOD

y primary outcome (continuous): school performance at the end of the school year (ranging from 1 to 5, higher values reflect better performance)

References

Pelham Jr, W. E., & Fabiano, G. A. (2008). Evidence-based psychosocial treatments for attention-deficit/hyperactivity disorder. Journal of Clinical Child & Adolescent Psychology, 37(1), 184-214.

Examples

```
data(adhd)
attach(adhd)
n = length(a1)
H1 = scale(cbind(o11, o12, o13, o14))
H2 = scale(cbind(H1, a1, H1*a1, r, o22, r*a1, o22*a1))
colnames(H2)[12] = "r*a1"
colnames(H2)[13] = "o22*a1"

fit_ql = ql(H=list(H1, H2), AA=list(a1,a2), RR=list(rep(0, n), y),
            pi=list(rep(0.5, n), rep(0.5,n)), K=2, m=3, lasso=TRUE)

c = 2^c(-3:3)
fit_owl = owl(H=list(H1, H2), AA=list(a1,a2), RR=list(rep(0, n), y),
              n=n, K=2, pi=list(rep(0.5, n), rep(0.5,n)), res.lasso = TRUE,
              loss="hinge", kernel="linear", augment=TRUE, c=c, m=3)
```

---

owl	*Integrated Outcome-weighted Learning for Estimating Optimal DTRs*

Description

This function implements a variety of outcome-weighted learning methods for estimating general K-stage DTRs. Different loss functions - SVM hinge loss, SVM ramp loss, binomial deviance loss, and L2 loss - can be adopted to solve the weighted classification problem at each stage. Augmentation in the outcomes is allowed to improve efficiency especially when there are multiple stages with a small sample size. Cross validation is conducted to choose the best tuning parameters if any.

Usage

```
owl(H, AA, RR, n, K, pi='estimated', res.lasso=TRUE, loss='hinge', kernel='linear',
augment=FALSE, c=2^(-2:2), sigma=c(0.03,0.05,0.07), s=2.^(-2:2), m=4)
```

Arguments

H	subject history information before treatment for the K stages. It can be a vector or a matrix when only baseline information is used in estimating the DTR; otherwise, it would be a list of length K. Please standardize all the variables in H to have mean 0 and standard deviation 1 before using H as the input. See details for how to construct H.
AA	observed treatment assignments for all subjects at the K stages. It is a vector if K=1, or a list of K vectors corresponding to the K stages.
RR	observed reward outcomes for all subjects at the K stages. It is a vector if K=1, or a list of K vectors corresponding to the K stages.
n	sample size, number of subjects in the dataset

K	number of stages
pi	treatment assignment probabilities of the observed treatments for all subjects at the K stages. It is a vector if K=1, or a list of K vectors corresponding to the K stages. It can be a user specified input if the treatment assignment probabilities are known. The default is pi="estimated", that is we estimate the treatment assignment probabilities based on lasso-penalized logistic regressions with H_k being the predictors at each stage k.
res.lasso	whether or not to use lasso penalty in the regression to take residuals for constructing the weights. The default is res.lasso=TRUE.
loss	loss function for sovling the weighted classification problem at each stage. The options are "hinge", "ramp", "logit", "logit.lasso", "l2", "l2.lasso". "hinge" and "ramp" are for the SVM hinge loss and SVM ramp loss. "logit" and "logit.lasso" are for the binomial deviance loss used in the logistic regression, where lasso penalty is applied under "logit.lasso". "l2" and "l2.lasso" are for the L2 or square loss, where lasso penalty is applied under "l2.lasso". The default is loss="hinge".
kernel	kernel function to use under SVM hinge loss or SVM ramp loss. "linear" and "rbf" kernel are implemented under SVM hinge loss; "linear" kernel is implemented under SVM ramp loss. The default is kernel="linear".
augment	whether or not to use augmented outcomes at each stage. Augmentation is recommended when there are multiple stages and the sample size is small. The default is augment=FALSE.
c	a vector specifies the values of the regularization parameter C for tuning under SVM hinge loss or SVM ramp loss. The default is c=2^(-2:2). In practice, a wider range of c can be specified based on the data.
sigma	a vector specifies the values of the positive parameter σ in the RBF kernel for tuning under SVM hinge loss, i.e., when loss="hinge" and kernel="rbf". The default is sigma=c(0.03,0.05,0.07). In practice, a wider range of sigma can be specified based on the data.
s	a vector specifies the values of the slope parameter in the SVM ramp loss for tuning, i.e., when loss="ramp" and kernel="linear". The default is c=2^(-2:2). In practice, a wider range of s can be specified based on the data.
m	number of folds in the m-fold cross validation for choosing the tuning parameters c, sigma or s. It is also used for choosing the tuning parameter of the lasso penalty when res.lasso=T, loss="logit.lasso" or loss="l2.lasso" is specified. The default is m=4.

Details

A patient's history information prior to the treatment at stage k can be constructed recursively as $H_k = (H_{k-1}, A_{k-1}, R_{k-1}, X_k)$ with $H_1 = X_1$, where X_k is subject-specific variables collected at stage k just prior to the treatment, A_k is the treatment at stage k, and R_k is the outcome observed post the treatment at stage k. Higher order or interaction terms can also be easily incorporated in H_k, e.g., $H_k = (H_{k-1}, A_{k-1}, R_{k-1}, X_k, H_{k-1}A_{k-1}, R_{k-1}A_{k-1}, X_kA_{k-1})$.

Value

A list of results is returned as an object. It contains the following attributes:

stage1	a list of stage 1 results, ...
stageK	a list of stage K results
valuefun	overall empirical value function under the estimated DTR
benefit	overall empirical benefit function under the estimated DTR
pi	treatment assignment probabilities of the observed treatments for each subject at the K stages. It is a list of K vectors. If pi='estimated' is specified as input, the estimated treatment assignment probabilities from lasso-penalized logistic regressions will be returned.
type	object type corresponding to the specified loss and kernel

In each stage's result, a list is returned which consists of

beta0	estimated coefficient of the intercept in the decision function
beta	estimated coefficients of H_k in the decision function. It's not returned with RBF kernel under SVM hinge loss.
fit	fitted decision function for each subject
probability	estimated probability that treatment 1 (vs. -1) is the optimal treatment for each subject in the sample. It's calculated by exp(fit)/(1 + exp(fit)).
treatment	the estimated optimal treatment for each subject
c	the best regularization parameter C in SVM hinge loss or SVM ramp loss, chosen from the values specified in c via cross validation
sigma	the best parameter σ in the RBF kernel, chosen from the values specified in sigma via cross validation
s	the best slope parameter s in the ramp loss, chosen from the values specified in s via cross validation.
iter	number of iterations under SVM ramp loss
alpha1	the solution to the Lagrangian dual problem under SVM hinge loss or SVM ramp loss. It is used for constructing the decision function on the new sample.
H	the input H, returned only under SVM hinge loss with RBF kernel. It is used for constructing the RBF kernel on the new sample.

Author(s)

Yuan Chen, Ying Liu, Donglin Zeng, Yuanjia Wang

Maintainer: Yuan Chen <yc3281@columbia.edu><irene.yuan.chen@gmail.com>

References

Liu, Y., Wang, Y., Kosorok, M., Zhao, Y., & Zeng, D. (2014). Robust hybrid learning for estimating personalized dynamic treatment regimens. arXiv preprint. arXiv, 1611.

Liu, Y., Wang, Y., Kosorok, M., Zhao, Y., & Zeng, D. (2018). Augmented Outcome-weighted Learning for Estimating Optimal Dynamic Treatment Regimens. Statistics in Medicine. In press.

Zhao, Y., Zeng, D., Rush, A. J., & Kosorok, M. R. (2012). Estimating individualized treatment rules using outcome weighted learning. Journal of the American Statistical Association, 107(499), 1106-1118.

Zhao, Y. Q., Zeng, D., Laber, E. B., & Kosorok, M. R. (2015). New statistical learning methods for estimating optimal dynamic treatment regimes. Journal of the American Statistical Association, 110(510), 583-598.

See Also

predict.owl, sim_Kstage, ql

Examples

```
# simulate 2-stage training and test sets
n_train = 100
n_test = 500
n_cluster = 10
pinfo = 10
pnoise = 20

train = sim_Kstage(n_train, n_cluster, pinfo, pnoise, K=2)
H1_train = scale(train$X)
H2_train = scale(cbind(H1_train, train$A[[1]], H1_train * train$A[[1]]))
pi_train = list(rep(0.5, n_train), rep(0.5, n_train))

test = sim_Kstage(n_test, n_cluster, pinfo, pnoise, train$centroids, K=2)
H1_test = scale(test$X)
H2_test = scale(cbind(H1_test, test$A[[1]], H1_test * train$A[[1]]))
pi_test = list(rep(0.5, n_test), rep(0.5, n_test))

# estimate DTR with owl on the training sample
owl_train = owl(H=list(H1_train, H2_train), AA=train$A, RR=train$R, n=n_train, K=2, pi=pi_train,
    loss='hinge', augment=TRUE, m=3)
owl_train$stage1$beta
owl_train$stage1$treatment
owl_train$valuefun

# apply the estimated DTR to the test sample
owl_test = predict(owl_train, H=list(H1_test, H2_test), AA=test$A, RR=test$R, K=2, pi=pi_test)
owl_test$treatment
owl_test$valuefun
```

---

predict.owl	*Predict from a Fitted "owl" Object*

Description

This function serves two purposes from a fitted "owl" object. It can recommend treatments for a new independent sample with partial or full subject features observed up to a certain stage. If subject feautures, treatment assignments and outcomes are fully observed in the new sample, this function can also evaluate the estimated DTR on this new sample, returning the empirical value function and benefit function.

Usage

```
## S3 method for class 'owl'
predict(object, H, AA=NULL, RR=NULL, K, pi=NULL, ...)
```

Arguments

object	fitted "owl" object
H	subject history information before treatment at the K stages for all subjects in the new sample. It should be constructed the same way as the H in fitting the owl object. See [owl](#) for how to construct H. Partial history information is allowed - when first j (j<=K) stages of H is specified, the first j stage treatments will be recommended.
AA	observed treatment assignments at the K stages for all subjects in the new sample. It is a vector if K=1, or a list of K vectors corresponding to the K stages. If not specified, treatments will be recommended for the new sample instead of DTR evaluation. The default is AA=NULL.
RR	observed outcomes at the K stages for all subjects in the new sample. It is a vector if K=1, or a list of K vectors corresponding to the K stages. If not specified, treatments will be recommended for the new sample instead of DTR evaluation. The default is RR=NULL.
K	number of stages of H observed in the new sample
pi	treatment assignment probabilities of the observed treatments at the K stages for all subjects in the new sample. It is a vector if K=1, or a list of K vectors corresponding to the K stages. It can be unspecified if one is only interested in treatment recommendations for the new sample. If both AA and RR are specified while pi is not specified, we will estimate the treatment assignment probabilities based on lasso-penalized logistic regressions with predictors being H_k at each stage k. The default is pi=NULL.
...	further arguments passed to or from other methods.

Value

fit	fitted decision functions at the K stages for each subject in the new sample. It is a list of K vectors.
probability	estimated probability that treatment 1 (vs. -1) is the optimal treatment at each stage for each subject in the new sample. It's calculated by $\exp(\text{fit})/(1 + \exp(\text{fit}))$. It is a list of K vectors.
treatment	recommennded optimal treatments at the K stages for each subject in the new sample. It is a list of K vectors.
valuefun	overall empirical value function under the fitted DTR evaluated on the new sample. It is returned only when AA and RR are fully specified for the K stages.
benefit	overall empirical benefit function under the estimated DTR evaluated on the new sample. It is returned only when AA and RR are fully specified for the K stages.
pi	treatment assignment probabilities of the assigned treatments at the K stages for each subject in the new sample. If pi is not specified but H and AA are specified for the K stages, the estimated treatment assignment probabilities from lasso-penalized logistic regressions with H_k being the predictors at each stage k will be returned.

Author(s)

Yuan Chen, Ying Liu, Donglin Zeng, Yuanjia Wang

Maintainer: Yuan Chen <yc3281@columbia.edu><irene.yuan.chen@gmail.com>

See Also

[owl](owl), sim_Kstage, [ql](ql)

Examples

```
# simulate 2-stage training and test sets
n_train = 100
n_test = 500
n_cluster = 10
pinfo = 10
pnoise = 20

train = sim_Kstage(n_train, n_cluster, pinfo, pnoise, K=2)
H1_train = scale(train$X)
H2_train = scale(cbind(H1_train, train$A[[1]], H1_train * train$A[[1]]))
pi_train = list(rep(0.5, n_train), rep(0.5, n_train))

test = sim_Kstage(n_test, n_cluster, pinfo, pnoise, train$centroids, K=2)
H1_test = scale(test$X)
H2_test = scale(cbind(H1_test, test$A[[1]], H1_test * train$A[[1]]))
pi_test = list(rep(0.5, n_test), rep(0.5, n_test))

# estimate DTR with owl on the training sample
owl_train = owl(H=list(H1_train, H2_train), AA=train$A, RR=train$R, n=n_train, K=2,
```

```
        pi=pi_train, loss='hinge', augment=TRUE, m=3)

# evaluate the DTR when full information are observed on the new sample
owl_test = predict(owl_train, H=list(H1_test, H2_test), AA=test$A, RR=test$R, K=2)
owl_test$treatment
owl_test$valuefun
owl_test$pi

# recommned the first-stage treatments only
owl_test2 = predict(owl_train, H=H1_test, K=1)
owl_test$treatment
```

---

predict.ql   *Predict from a Fitted "ql" Object*

Description

This function serves two purposes from a fitted "ql" object. It can recommend treatments for a new
independent sample with partial or full subject features observed up to a certain stage. If subject
features, treatment assignments and outcomes are fully observed in the new sample, this function
can also evaluate the fitted DTR on this new sample, returning the empirical value function and
benefit function.

Usage

```
## S3 method for class 'ql'
predict(object, H, AA=NULL, RR=NULL, K, pi=NULL, Qopt=FALSE, Qfit=FALSE, ...)
```

Arguments

object	fitted "ql" object
H	subject history information before treatment at the K stages for all subjects in the new sample. It should be constructed the same way as the H in fitting the ql object. See [ql](#) for how to construct H. Partial history information is allowed - when first j (j<=K) stages of H is specified, the first j stage treatments will be recommended.
AA	observed treatment assignments at the K stages for all subjects in the new sample. It is a vector if K=1, or a list of K vectors corresponding to the K stages. If not specified, treatments will be recommended for the new sample instead of DTR evaluation. The default is AA=NULL.
RR	observed outcomes at the K stages for all subjects in the new sample. It is a vector if K=1, or a list of K vectors corresponding to the K stages. If not specified, treatments will be recommended for the new sample instead of DTR evaluation. The default is RR=NULL.
K	number of stages of H observed in the new sample

pi   treatment assignment probabilities of the observed treatments at the K stages
                      for all subjects in the new sample. It is a vector if K=1, or a list of K vectors
                      corresponding to the K stages. It can be unspecified if one is only interested in
                      treatment recommendations for the new sample. If both AA and RR are specified
                      while pi is not specified, we will estimate the treatment assignment probabilities
                      based on lasso-penalized logistic regressions with predictors being H_k at each
                      stage k. The default is pi=NULL.

Qopt   whether to output the predicted optimal Q-function. The default is Qfun=FALSE.

Qfit   whether to output the estimated Q-function under the observed treatments. The
                      default is fitted=FALSE.

...   further arguments passed to or from other methods.

Value

treatment   recommennded optimal treatments at the K stages for each subject in the new
                      sample. It is a list of K vectors. If no tailoring variables are learned in the "ql"
                      object, treatments will be assigned randomly with equal probability.

valuefun   overall empirical value function under the fitted DTR evaluated on the new sam-
                      ple. It is returned only when AA and RR are fully specified for the K stages.

benefit   overall empirical benefit function under the estimated DTR evaluated on the new
                      sample. It is returned only when AA and RR are fully specified for the K stages.

pi   treatment assignment probabilities of the assigned treatments at the K stages for
                      each subject in the new sample. If pi is not specified but H and AA are specified
                      for the K stages, the estimated treatment assignment probabilities from lasso-
                      penalized logistic regressions with H_k being the predictors at each stage k will
                      be returned.

Q   the predicted optimal Q-function if Qfun=TRUE.

fitted   the estimated Q-function under the observed treatment if fitted=TRUE.

Author(s)

Yuan Chen, Ying Liu, Donglin Zeng, Yuanjia Wang

Maintainer: Yuan Chen <yc3281@columbia.edu><irene.yuan.chen@gmail.com>

See Also

[ql](#), [sim_Kstage](#), [owl](#)

Examples

```
# simulate 2-stage training and test sets
n_train = 100
n_test = 500
n_cluster = 10
pinfo = 10
pnoise = 20
```

```
train = sim_Kstage(n_train, n_cluster, pinfo, pnoise, K=2)
H1_train = scale(train$X)
H2_train = scale(cbind(H1_train, train$A[[1]], H1_train * train$A[[1]]))
pi_train = list(rep(0.5, n_train), rep(0.5, n_train))

test = sim_Kstage(n_test, n_cluster, pinfo, pnoise, train$centroids, K=2)
H1_test = scale(test$X)
H2_test = scale(cbind(H1_test, test$A[[1]], H1_test * train$A[[1]]))
pi_test = list(rep(0.5, n_test), rep(0.5, n_test))

# estimate DTR with ql on the training sample
ql_train = ql(H=list(H1_train, H2_train), AA=train$A, RR=train$R, K=2, pi=pi_train, m=3)

# evaluate the DTR when full information are observed on the new sample
ql_test = predict(ql_train, H=list(H1_test, H2_test), AA=test$A, RR=test$R, K=2)
ql_test$treatment
ql_test$valuefun
ql_test$pi

# recommned the first-stage treatments only
ql_test2 = predict(ql_train, H=H1_test, K=1)
ql_test2$treatment
```

---

ql   *Q-learning for Estimating Optimal DTRs*

Description

This function implements Q-learning for estimating general K-stage DTRs. Lasso penalty can be applied for variable selection at each stage.

Usage

```
ql(H, AA, RR, K, pi='estimated', lasso=TRUE, m=4)
```

Arguments

H	subject history information before treatment for all subjects at the K stages. It can be a vector or a matrix when only baseline information is used in estimating the DTR; otherwise, it would be a list of length K. Please standardize all the variables in H to have mean 0 and standard deviation 1 before using H as the input. See details for how to construct H.
AA	observed treatment assignments for all subjects at the K stages. It is a vector if K=1, or a list of K vectors corresponding to the K stages.
RR	observed reward outcomes for all subjects at the K stages. It is a vector if K=1, or a list of K vectors corresponding to the K stages.
K	number of stages

pi	treatment assignment probabilities of the observed treatments for all subjects at the K stages. It is a vector if K=1, or a list of K vectors corresponding to the K stages. It can be a user specified input if the treatment assignment probabilities are known. The default is pi="estimated", that is we estimate the treatment assignment probabilities based on lasso-penalized logistic regressions with H_k being the predictors at each stage k.
lasso	specifies whether to add lasso penalty at each stage when fitting the model. The default is lasso=TRUE.
m	number of folds in the m-fold cross validation. It is used when res.lasso=T is specified. The default is m=4.

Details

A patient's history information prior to the treatment at stage k can be constructed recursively as $H_k = (H_{k-1}, A_{k-1}, R_{k-1}, X_k)$ with $H_1 = X_1$, where X_k is subject-specific variables collected at stage k just prior to the treatment, A_k is the treatment at stage k, and R_k is the outcome observed post the treatment at stage k. Higher order or interaction terms can also be easily incorporated in H_k, e.g., $H_k = (H_{k-1}, A_{k-1}, R_{k-1}, X_k, H_{k-1}A_{k-1}, R_{k-1}A_{k-1}, X_kA_{k-1})$.

Value

A list of results is returned as an object. It contains the following attributes:

stage1	a list of stage 1 results, ...
stageK	a list of stage K results
valuefun	overall empirical value function under the estimated DTR
benefit	overall empirical benefit function under the estimated DTR
pi	treatment assignment probabilities of the assigned treatments for each subject at the K stages. If pi='estimated' is specified as input, the estimated treatment assignment probabilities from lasso-penalized logistic regressions will be returned.

In each stage's result, a list is returned which consists of

co	the estimated coefficients of $(1, H, A, H * A)$, the variables in the model at this stage
treatment	the estimated optimal treatment at this stage for each subject in the sample. If no tailoring variables are selected under lasso penalty, treatment will be assigned randomly with equal probability.
Q	the estimated optimal outcome increment from this stage to the end (the estimated optimal Q-function at this stage) for each subject in the sample

Author(s)

Yuan Chen, Ying Liu, Donglin Zeng, Yuanjia Wang

Maintainer: Yuan Chen <yc3281@columbia.edu><irene.yuan.chen@gmail.com>

References

Watkins, C. J. C. H. (1989). Learning from delayed rewards (Doctoral dissertation, University of Cambridge).

Qian, M., & Murphy, S. A. (2011). Performance guarantees for individualized treatment rules. Annals of statistics, 39(2), 1180.

See Also

predict.ql, sim_Kstage, owl

Examples

```
# simulate 2-stage training and test sets
n_train = 100
n_test = 500
n_cluster = 10
pinfo = 10
pnoise = 20

train = sim_Kstage(n_train, n_cluster, pinfo, pnoise, K=2)
H1_train = scale(train$X)
H2_train = scale(cbind(H1_train, train$A[[1]], H1_train * train$A[[1]]))
pi_train = list(rep(0.5, n_train), rep(0.5, n_train))

test = sim_Kstage(n_test, n_cluster, pinfo, pnoise, train$centroids, K=2)
H1_test = scale(test$X)
H2_test = scale(cbind(H1_test, test$A[[1]], H1_test * train$A[[1]]))
pi_test = list(rep(0.5, n_test), rep(0.5, n_test))

ql_train = ql(H=list(H1_train, H2_train), AA=train$A, RR=train$R, K=2, pi=pi_train, m=3)

ql_test = predict(ql_train, H=list(H1_test, H2_test), AA=test$A, RR=test$R, K=2, pi=pi_test)
```

---

sim_Kstage *Simulate a K-stage Sequential Multiple Assignment Randomized Trial (SMART) data*

Description

This function simulates a K-stage SMART data with (pinfo + pnoise) baseline variables from a multivariate Gaussian distribution. The pinfo variables have variance 1 and pairwise correlation 0.2; the pnoise variables have mean 0 and are uncorrelated with each other and with the pinfo variables.

Subjects are from n_cluster latent groups with equal sizes, and these n_cluster groups are characterized by their differentiable means in the pinfo feature variables. Each latent group has its own optimal treatment sequence, where the optimal treatment for subjects in group g at stage k is generated as $A^* = 2([g/(2k-1)] \bmod 2)$ - 1. The assigned treatment group (1 or -1) for each subject at each stage is randomly generated with equal probability. The primary outcome is observed only at the end of the trial, which is generated as $R = \sum_{k=1}^{K} A_k A_k^* + N(0,1)$.

Usage

```
sim_Kstage (n, n_cluster, pinfo, pnoise, centroids=NULL, K)
```

Arguments

n	sample size, should be a multiple of n_cluster.
n_cluster	number of latent groups
pinfo	number of informative baseline variables
pnoise	number of non-informative baseline variables
centroids	centroids of the pinfo variables for the n_cluster groups. It is a matrix of dimension n_cluster by pinfo. It's used as the means of the multivariate Gaussians to generate the pinfo variables for the n_cluster groups. For a training set, do not assign centroids, the centroids are generated randomly from N(0,5) by the function. For a test set, one should assign the same set of centroids as the training set.
K	number of stages.

Value

X	baseline variables. It is a matrix of dimension n by (pinfo + pnoise).
A	treatment assigments for the K-stages. It is a list of K vectors.
R	outcomes of the K-stages. It is a list of K vectors. In this simulation setting, no intermediate outcomes are observed, so the first K-1 vectors are vectors of 0.
optA	optimal treatments for the K-stages. It is a list of K vectors.
centroids	centroids of the pinfo variables for the n_cluster groups. It is a matrix of dimension n_cluster by pinfo.

Author(s)

Yuan Chen, Ying Liu, Donglin Zeng, Yuanjia Wang

Maintainer: Yuan Chen <yc3281@columbia.edu><irene.yuan.chen@gmail.com>

See Also

[owl](#), [ql](#)

Examples

```
n_train = 100
n_test = 500
n_cluster = 10
pinfo = 10
pnoise = 20

# simulate a 2-stage training set
train = sim_Kstage(n_train, n_cluster, pinfo, pnoise, K=2)
```

```
# simulate an independent 2-stage test set with the same centroids of the training set
test = sim_Kstage(n_test, n_cluster, pinfo, pnoise, train$centroids, K=2)
```

Index