Package 'sgee'

October 14, 2022

Type Package

Title Stagewise Generalized Estimating Equations

Version 0.6-0

Date 2018-01-08

Description Stagewise techniques implemented with Generalized Estimating Equations to handle individual, group, bi-level, and interaction selection. Stagewise approaches start with an empty model and slowly build the model over several iterations, which yields a 'path' of candidate models from which model selection can be performed. This 'slow brewing' approach gives stagewise techniques a unique flexibility that allows simple incorporation of Generalized Estimating Equations; see Vaughan, G., Aseltine, R., Chen, K., Yan, J., (2017) <doi:10.1111/biom.12669> for details.

Author Gregory Vaughan [aut, cre],
Kun Chen [ctb], Jun Yan [ctb]

Maintainer Gregory Vaughan <gvaughan@bentley.edu>

Imports mvtnorm, copula, stats, utils

Depends R (>= 3.0.0)

License GPL (>= 3)

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-08 18:34:38 UTC

R topics documented:

`sgee-package` *sgee: Stagewise Generalized Estimating Equations*

Description

Provides functions to perform Boosting / Functional Gradient Descent / Forward Stagewise regression with grouped covariates setting using Generalized Estimating Equations.

Details

Package:	sgee
Type:	Package
Version:	0.6-0
Date:	2018-01-08
License:	GPL (>= 3)

sgee provides several stagewise regression approaches that are designed to address variable selection with grouped covariates in the context of Generalized Estimating Equations. Given a response and design matrix stagewise techniques perform a sequence of small learning steps wherein a subset of the covariates are selected as being the most important at that iteration and are then subsequently updated by a small amount, epsilon. different techniques this optimal update in different ways that achieve different structural goals (i.e. groups of covariates are fully included or not).

The resulting path can then be analyzed to determine an optimal model along the path of coefficient estimates. The `analyzeCoefficientPath` function provides such functionality based on various possible metrics, primarily focused on the Mean Squared Error. Furthermore, the `plot.sgee` function can be used to examine the path of coefficient estimates versus the iteration number, or some desired penalty.

Author(s)

Gregory Vaughan [aut, cre], Kun Chen [ctb], Jun Yan [ctb]

Maintainer: Gregory Vaughan <gregory.vaughan@uconn.edu>

References

Vaughan, G., Aseltine, R., Chen, K., Yan, J., (2017). Stagewise Generalized Estimating Equations with Grouped Variables. Biometrics 73, 1332-1342. URL: http://dx.doi.org/10.1111/biom.12669, doi:10.1111/biom.12669.

Vaughan, G., Aseltine, R., Chen, K., Yan, J., (2017). Efficient interaction selection for clustered data via stagewise generalized estimating equations. Department of Statistics, University of Connecticut. Technical Report.

Wolfson, J. (2011). EEBoost: A general method for prediction and variable selection based on estimating equations. Journal of the American Statistical Association 106, 296–305.

Tibshirani, R. J. (2015). A general framework for fast stagewise algorithms. Journal of Machine Learning Research 16, 2543–2588.

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2013). A sparse-group lasso. Journal of Computational and Graphical Statistics 22, 231–245.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.

Liang, K.-Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. Biometrika 73, 13–22.

Examples

```
#####################
## Generate test data
#####################

## Initialize covariate values
p <- 50
beta <- c(rep(2.4,5),
          c(1.2, 0, 1.6, 0, .4),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 5
numGroups <- length(beta)/groupSize


generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         numGroups = numGroups,
                         groupSize = groupSize,
                         groupRho = 0.3,
                         beta = beta,
                         family = gaussian(),
                         intercept = 0)


coefMat1 <- hisee(y = generatedData$y, x = generatedData$x,
                  family = gaussian(),
                  clusterID = generatedData$clusterID,
```

```
                  groupID = generatedData$groupID,
                  corstr="exchangeable",
                  control = sgee.control(maxIt = 100, epsilon = 0.2))

## interceptLimit allows for compatibility with older R versions
coefMat2 <- bisee(y = generatedData$y, x = generatedData$x,
                  family = gaussian(),
                  clusterID = generatedData$clusterID,
                  groupID = generatedData$groupID,
                  corstr="exchangeable",
                  control = sgee.control(maxIt = 100, epsilon = 0.2,
                                         interceptLimit = 10),
                  lambda1 = .5,
                  lambda2 = .5)


par(mfrow = c(2,1))
plot(coefMat1)
plot(coefMat2)
```

---

bisee *Bi-Level Stagewise Estimating Equations Implementation*

Description

Function to perform BiSEE, a Bi-Level Boosting / Functional Gradient Descent / Forward Stagewise regression in the grouped covariates setting using Generalized Estimating Equations

Usage

```
bisee(y, ...)

## S3 method for class 'formula'
bisee(formula, data = list(), clusterID, waves = NULL,
  lambda1, lambda2 = 1 - lambda1, contrasts = NULL, subset, ...)

## Default S3 method:
bisee(y, x, waves = NULL, lambda1, lambda2 = 1 - lambda1,
  ...)

## S3 method for class 'fit'
bisee(y, x, family, clusterID, waves = NULL, groupID,
  corstr = "independence", alpha = NULL, lambda1 = 0.5, lambda2 = 1 -
  lambda1, intercept = TRUE, offset = 0, control = sgee.control(maxIt =
  200, epsilon = 0.05, stoppingThreshold = min(length(y), ncol(x)) - intercept,
  undoThreshold = 0.005), standardize = TRUE, verbose = FALSE, ...)

gsee(y, x, family, clusterID, waves = NULL, groupID = 1:ncol(x),
```

```
corstr = "independence", alpha = NULL, offset = 0, intercept = TRUE,
control = sgee.control(maxIt = 200, epsilon = 0.05, stoppingThreshold =
min(length(y), ncol(x)) - intercept, undoThreshold = 0.005),
standardize = TRUE, verbose = FALSE, ...)
```

Arguments

y	Vector of response measures that corresponds with modeling family given in 'family' parameter. y is assumed to be the same length as clusterID and is assumed to be organized into clusters as dictated by clusterID.
...	Not currently used
formula	Object of class 'formula'; a symbolic description of the model to be fitted
data	Optional data frame containing the variables in the model.
clusterID	Vector of integers that identifies the clusters of response measures in y. Data and clusterID are assumed to 1) be of equal lengths, 2) sorted so that observations of a cluster are in contiguous rows, and 3) organized so that clusterID is a vector of consecutive integers.
waves	An integer vector which identifies components in clusters. The length of waves should be the same as the number of observations. waves is automatically generated if none is supplied, but when using subset parameter, the waves parameter must be provided by the user for proper calculation.
lambda1	Mixing parameter used to indicate weight of L_2 Norm (group selection). While not necessary, lambda1 and lambda2 are best set to sum to 1 as only the weights relative to each other matter. Default value is set to .5.
lambda2	Mixing parameter used to indicate weight of L_1 Norm (individual selection). While not necessary, lambda1 and lambda2 are best set to sum to 1 as only the weights relative to each other matter. Default value is set to 1-lambda1.
contrasts	An optional list provided when using a formula. similar to contrasts from glm. See the contrasts.arg of model.matrix.default.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
x	Design matrix of dimension length(y) x nvar, the number of variables, where each row is represents an observation of predictor variables.
family	Modeling family that describes the marginal distribution of the response. Assumed to be an object such as gaussian() or poisson().
groupID	Vector of integeres that identifies the groups of the covariates/coefficients (i.e. the columns of x). x and groupID are assumed 1) to be of corresponding dimension, (i.e. ncol(x) == length(groupID)), 2) sorted so that groups of covariates are in contiguous columns, and 3) organized so that groupID is a vector of consecutive integers.
corstr	A character string indicating the desired working correlation structure. The following are implemented : "independence" (default value), "exchangeable", and "ar1".
alpha	An initial guess for the correlation parameter value between -1 and 1 . If left NULL (the default), the initial estimate is 0.

intercept	Binary value indicating where an intercept term is to be included in the model for estimation. Default is to include an intercept.
offset	Vector of offset value(s) for the linear predictor. offset is assumed to be either of length one, or of the same length as y. Default is to have no offset.
control	A list of parameters used to contorl the path generation process; see sgee.control.
standardize	A logical parameter that indicates whether or not the covariates need to be standardized before fitting. If standardized before fitting, the unstandardized path is returned as the default, with a standardizedPath and standardizedX included separately. Default value is TRUE.
verbose	Logical parameter indicating whether output should be produced while bisee is running. Default value is FALSE.

Details

Function to implement BiSEE, a stagewise regression approach that is designed to perform bi-level selection in the context of Generalized Estimating Equations. Given a response y and a design matrix x (excluding intercept) BiSEE generates a path of stagewise regression estimates for each covariate based on the provided step size epsilon, and tuning parameters lambda1 and lambda2. When lambda1 == 0 or lambda2 == 0, the simplified versions of bisee called see and gsee, respectively, will be called.

The resulting path can then be analyzed to determine an optimal model along the path of coefficient estimates. The summary.sgee function provides such functionality based on various possible metrics, primarily focused on the Mean Squared Error. Furthermore, the plot.sgee function can be used to examine the path of coefficient estimates versus the iteration number, or some desired penalty.

bisee makes use of the function uniroot in the stats package. The extendInt parameter for uniroot is used, which may cause issues for older versions of R.

Value

Object of class sgee containing the path of coefficient estimates, the path of scale estimates, the path of correlation parameter estimates, the iteration at which BiSEE terminated, and initial regression values including x, y, codefamily, clusterID, groupID, offset, epsilon, and numIt.

Note

Function to execute BiSEE technique. Note that lambda1 and lambda2 are tuning parameters. Though it is advised to fix lambda1 + lambda2 = 1, this is not necessary. These parameters can be tuned using various approaches including cross validation.

Author(s)

Gregory Vaughan

References

Vaughan, G., Aseltine, R., Chen, K., Yan, J., (2017). Stagewise Generalized Estimating Equations with Grouped Variables. Biometrics 73, 1332-1342. URL: http://dx.doi.org/10.1111/biom.12669, doi:10.1111/biom.12669.

Wolfson, J. (2011). EEBoost: A general method for prediction and variable selection based on estimating equations. Journal of the American Statistical Association 106, 296–305.

Tibshirani, R. J. (2015). A general framework for fast stagewise algorithms. Journal of Machine Learning Research 16, 2543–2588.

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2013). A sparse-group lasso. Journal of Computational and Graphical Statistics 22, 231–245.

Examples

```
#####################
## Generate test data
#####################

## Initialize covariate values
p <- 50
beta <- c(rep(2,5),
          c(1, 0, 1.5, 0, .5),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 5
numGroups <- length(beta)/groupSize


generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         numGroups = numGroups,
                         groupSize = groupSize,
                         groupRho = 0.3,
                         beta = beta,
                         family = gaussian(),
                         intercept = 1)


## Perform Fitting by providing y and x values
coefMat1 <- bisee(y = generatedData$y, x = generatedData$x,
                 family = gaussian(),
                 clusterID = generatedData$clusterID,
                 groupID = generatedData$groupID,
                 corstr = "exchangeable",
                 control = sgee.control(maxIt = 50, epsilon = 0.5),
```

```
                  lambda1 = .5,
                  lambda2 = .5,
                  verbose = TRUE)


## Perform Fitting by providing formula and data
genDF <- data.frame(generatedData$y, generatedData$x)
names(genDF) <- c("Y", paste0("Cov", 1:p))
coefMat2 <- bisee(formula(genDF), data = genDF,
                  family = gaussian(),
                  subset = Y <1.5,
                  waves = rep(1:4, 50),
                  clusterID = generatedData$clusterID,
                  groupID = generatedData$groupID,
                  corstr = "exchangeable",
                  control = sgee.control(maxIt = 50, epsilon = 0.5),
                  lambda1 = 0.5,
                  lambda2 = 0.5,
                  verbose = TRUE)

par(mfrow = c(2,1))
plot(coefMat1)
plot(coefMat2)
```

`genCorMat` *Correlation Matrix Generator.*

Description

Function that generates a correlation matrix of a predefined type and size given appropriate correlation parameter(s), `rho`.

Usage

```
genCorMat(corstr = "independence", rho, maxClusterSize = 0)
```

Arguments

`corstr`	Structure of correlaiton matrix to be generated; 'independence', 'exchangeable', 'ar1', and 'unstructured' currently implemented.
`rho`	Correlation parameter; assumed to be of length 1 or `maxClusterSize * (maxClusterSize - 1) /2`.
`maxClusterSize`	size of the correlation matrix being generated.

Value

A correlation matrix of form matching corstr and of size maxClusterSize.

Note

Mostly intended for internal use, but could be useful to user. Therefore, the function is exported.

Author(s)

Gregory Vaughan

Examples

```
## Generates Correlation Matricies easily
## When corstr = "independence", the value of rho
## is irrelevant
mat1 <- genCorMat(corstr = "independence", rho = .1, maxClusterSize = 3)

## Exchangeable
mat2 <- genCorMat(corstr = "exchangeable", rho = .3, maxClusterSize = 2)

## AR-1
mat3 <- genCorMat(corstr = "ar1", rho = .4, maxClusterSize = 4)

## unstructured
mat3 <- genCorMat(corstr = "unstructured",
                  rho = c(.3,.2,.1),
                  maxClusterSize = 3)
```

---

genData *Response and Covariate Data Generation*

Description

Function to generate data that can be used to test Forward stagewise / Penalized Regression techniques. Currently marginally Gaussian and Poisson responses are possible.

Function is provided to allow the user simple data generation as sgee functions were designed for. Various parameters controlling aspects such as the response correlation, the covariate group structure, the marginal response distribution, and the signal to noise ratio for marginally gaussian responses are provided to allow a great deal of specificity over the kind of data that is generated.

Usage

```
genData(numClusters, clusterSize = 1, clusterRho = 0,
  clusterCorstr = "exchangeable", yVariance = NULL, xVariance = 1,
  numGroups = length(beta), groupSize = 1, groupRho = 0, beta = 0,
  numMainEffects = NULL, family = gaussian(), SNR = NULL, intercept = 0)
```

Arguments

numClusters	Number of clusters to be generated.
clusterSize	Size of each cluster.
clusterRho	Correlation parameter for response.
clusterCorstr	String indicating cluster Correlation structure. Parameter is fed to genCorMat, so all possible entries for genCorMat are allowed.
yVariance	Optional scalar value specifying the marginal response variance; overrides SNR.
xVariance	Scalar value indicating marginal variance of the covariates.
numGroups	Number of covariate groups to be generated. Default behavior is to generate groups of size 1 (effectively no groups). If covariate groups are desired, numGroups and groupSize must be given such that length(beta) equals numGroups * groupSize.
groupSize	Size of each group.
groupRho	Within group correlation parameter.
beta	Vector of coefficient values used to generate response.
numMainEffects	An integer indicating that the first numMainEffects terms in beta are to be treated as main effects and the remaining terms are pairwise interaction effects, which are in the same order as generated by model.matrix. Default value of NULL indicates no interaction terms are included. The use of numMainEffects overrides any covariate grouping structure provided by the user.
family	Marginal response family; currently gaussian() and poisson() are accepted.
SNR	Scalar value that allows fixing the signal to noise ratio as defined as the ratio of the (observed) variance in the linear predictor to the variance of the response conditioned on the covariates.
intercept	Scalar value indicating the true intercept value.

Value

List containing the generated response, y, the generated covariates, x, a vector identifying the responses clusters, clusterID, and a vector identifying the covariate groups, groupID.

Note

Function is ued to generate both the desired covariate structure and the desired response structure. To generate poisson responses, functions from the R package coupla are used.

Current implementation of interactions overwrites any previous grouping structure; that is the number of groups becomes p and the group sizes are set to 1.

Author(s)

Gregory Vaughan

Examples

```
## A resonse variance can be given,
dat1 <- genData(numClusters = 10,
                clusterSize = 4,
                clusterRho = .5,
                clusterCorstr = "exchangeable",
                yVariance = 1,
                xVariance = 1,
                numGroups = 5,
                groupSize = 4,
                groupRho = .5,
                beta = c(rep(1,8), rep(0,12)),
                family = gaussian(),
                intercept = 1)

## or the signal to noise ratio can be fixed
dat2 <- genData(numClusters = 10,
                clusterSize = 4,
                clusterRho = .5,
                clusterCorstr = "exchangeable",
                xVariance = 1,
                numGroups = 5,
                groupSize = 4,
                groupRho = .5,
                beta = c(rep(1,8), rep(0,12)),
                family = poisson(),
                SNR = 10,
                intercept = 1)
```

---

hisee *Hierarchical Stagewise Estimating Equations Implementation.*

Description

Function to perform HiSEE, a Bi-Level Boosting / Functional Gradient Descent / Forward Stagewise regression in the grouped covariates setting using Generalized Estimating Equations

Usage

```
hisee(y, ...)

## S3 method for class 'formula'
hisee(formula, data = list(), clusterID, waves = NULL,
  contrasts = NULL, subset, ...)

## Default S3 method:
```

```
hisee(y, x, waves = NULL, ...)

## S3 method for class 'fit'
hisee(y, x, family, clusterID, waves = NULL,
  groupID = 1:ncol(x), corstr = "independence", alpha = NULL,
  intercept = TRUE, offset = 0, control = sgee.control(maxIt = 200,
  epsilon = 0.05, stoppingThreshold = min(length(y), ncol(x)) - intercept,
  undoThreshold = 0), standardize = TRUE, verbose = FALSE, ...)
```

Arguments

y	Vector of response measures that corresponds with modeling family given in 'family' parameter. 'y' is assumed to be the same length as 'clusterID' and is assumed to be organized into clusters as dictated by 'clusterID'.
...	Not currently used
formula	Object of class 'formula'; a symbolic description of the model to be fitted
data	Optional data frame containing the variables in the model.
clusterID	Vector of integers that identifies the clusters of response measures in 'y'. Data and 'clusterID' are assumed to 1) be of equal lengths, 2) sorted so that observations of a cluster are in contiguous rows, and 3) organized so that 'clusterID' is a vector of consecutive integers.
waves	An integer vector which identifies components in clusters. The length of waves should be the same as the number of observations. waves is automatically generated if none is supplied, but when using subset parameter, the waves parameter must be provided by the user for proper calculation.
contrasts	An optional list provided when using a formula. similar to contrasts from glm. See the contrasts.arg of model.matrix.default.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
x	Design matrix of dimension length(y) x nvars where each row is represents an obersvation of predictor variables. Assumed to be scaled.
family	Modeling family that describes the marginal distribution of the response. Assumed to be an object such as 'gaussian()' or 'poisson()'
groupID	Vector of integeres that identifies the groups of the covariates/coefficients (i.e. the columns of 'x'). 'x' and 'groupID' are assumed 1) to be of corresponding dimension, (i.e. ncol(x) == length(groupID)), 2) sorted so that groups of covariates are in contiguous columns, and 3) organized so that 'groupID' is a vector of consecutive integers.
corstr	A character string indicating the desired working correlation structure. The following are implemented : "independence" (default value), "exchangeable", and "ar1".
alpha	An initial guess for the correlation parameter value between -1 and 1 . If left NULL (the default), the initial estimate is 0.
intercept	Binary value indicating where an intercept term is to be included in the model for estimation. Default is to include an intercept.

offset	Vector of offset value(s) for the linear predictor. 'offset' is assumed to be either of length one, or of the same length as 'y'. Default is to have no offset.
control	A list of parameters used to contorl the path generation process; see sgee.control.
standardize	A logical parameter that indicates whether or not the covariates need to be standardized before fitting. If standardized before fitting, the unstandardized path is returned as the default, with a standardizedPath and standardizedX included separately. Default value is TRUE.
verbose	Logical parameter indicating whether output should be produced while hisee is running. Default value is FALSE.

Details

Function to implement HiSEE, a stagewise regression approach that is designed to perform hierarchical selection in the context of Generalized Estimating Equations. Given A response Y, design matrix X (excluding intercept) HiSEE generates a path of stagewise regression estimates for each covariate based on the provided step size epsilon. First an optimal group of covariates is identified, and then an optimal covariate within that group is selected and then updated in each iterative step.

The resulting path can then be analyzed to determine an optimal model along the path of coefficient estimates. The summary.sgee function provides such functionality based on various possible metrics, primarily focused on the Mean Squared Error. Furthermore, the plot.sgee function can be used to examine the path of coefficient estimates versus the iteration number, or some desired penalty.

Value

Object of class 'sgee' containing the path of coefficient estimates, the path of scale estimates, the path of correlation parameter estimates, and the iteration at which HiSEE terminated, and initial regression values including x, y, codefamily, clusterID, groupID, offset, epsilon, and numIt.

Note

Function to execute HiSEE Technique. Functionally equivalent to SEE when all elements in groupID are unique.

Author(s)

Gregory Vaughan

References

Vaughan, G., Aseltine, R., Chen, K., Yan, J., (2017). Stagewise Generalized Estimating Equations with Grouped Variables. Biometrics 73, 1332-1342. URL: http://dx.doi.org/10.1111/biom.12669, doi:10.1111/biom.12669.

Wolfson, J. (2011). EEBoost: A general method for prediction and variable selection based on estimating equations. Journal of the American Statistical Association 106, 296–305.

Tibshirani, R. J. (2015). A general framework for fast stagewise algorithms. Journal of Machine Learning Research 16, 2543–2588.

Examples

```
#####################
## Generate test data
#####################

## Initialize covariate values
p <- 50
beta <- c(rep(2,5),
          c(1, 0, 1.5, 0, .5),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 5
numGroups <- length(beta)/groupSize


generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         numGroups = numGroups,
                         groupSize = groupSize,
                         groupRho = 0.3,
                         beta = beta,
                         family = gaussian(),
                         intercept = 1)

## Perform Fitting by providing y and x values
coefMat1 <- hisee(y = generatedData$y, x = generatedData$x,
                  family = gaussian(),
                  clusterID = generatedData$clusterID,
                  groupID = generatedData$groupID,
                  corstr="exchangeable",
                  control = sgee.control(maxIt = 50, epsilon = 0.5))

## Perform Fitting by providing formula and data
genDF <- data.frame(generatedData$y, generatedData$x)
names(genDF) <- c("Y", paste0("Cov", 1:p))
coefMat2 <- hisee(formula(genDF), data = genDF,
                  family = gaussian(),
                  subset = Y<1,
                  waves = rep(1:4, 50),
                  clusterID = generatedData$clusterID,
                  groupID = generatedData$groupID,
                  corstr="exchangeable",
                  control = sgee.control(maxIt = 50, epsilon = 0.5))

par(mfrow = c(2,1))
plot(coefMat1)
plot(coefMat2)
```

isee *Interaction stagewise estimating equations*

Description

Perform model selection with clustered data while considering interaction terms using one of two stagewise methods. The first (ACTS) uses an active set approach in which interaction terms are only considered for a given update if the corresponding main effects have already been added to the model. The second approach (HiLa) approximates the regularized path for hierarchical lasso with Generalized Estimating Equations. In this second approach, the model hierarchy is guaranteed in each individual step, thus ensuring the desired hierarchy throughout the path.

Usage

```
isee(y, ...)

## S3 method for class 'formula'
isee(formula, data = list(), clusterID, waves = NULL,
  interactionID = NULL, contrasts = NULL, subset, method = "ACTS", ...)

## Default S3 method:
isee(y, x, waves = NULL, interactionID, method = "ACTS",
  ...)

acts.fit(y, x, interactionID, family, clusterID, waves = NULL,
  corstr = "independence", alpha = NULL, intercept = TRUE, offset = 0,
  control = sgee.control(maxIt = 200, epsilon = 0.05, stoppingThreshold =
  min(length(y), ncol(x)) - intercept, undoThreshold = 0), standardize = TRUE,
  verbose = FALSE, ...)

hila.fit(y, x, interactionID, family, clusterID, waves = NULL,
  corstr = "independence", alpha = NULL, intercept = TRUE, offset = 0,
  control = sgee.control(maxIt = 200, epsilon = 0.05, stoppingThreshold =
  min(length(y), ncol(x)) - intercept, undoThreshold = 0.005),
  standardize = TRUE, verbose = FALSE, ...)
```

Arguments

y	Vector of response measures that corresponds with modeling family given in 'family' parameter. 'y' is assumed to be the same length as 'clusterID' and is assumed to be organized into clusters as dictated by 'clusterID'.
...	Not currently used
formula	Object of class 'formula'; a symbolic description of the model to be fitted
data	Optional data frame containing the variables in the model.

clusterID	Vector of integers that identifies the clusters of response measures in 'y'. Data and 'clusterID' are assumed to 1) be of equal lengths, 2) sorted so that observations of a cluster are in contiguous rows, and 3) organized so that 'clusterID' is a vector of consecutive integers.
waves	An integer vector which identifies components in clusters. The length of waves should be the same as the number of observations. waves is automatically generated if none is supplied, but when using subset parameter, the waves parameter must be provided by the user for proper calculation.
interactionID	A (p^2+p)/2 x 2 matrix of interaction IDs. Main effects have the same (unique) number in both columns for their corresponding row. Interaction effects have each of their corresponding main effects in the two columns. it is assumed that main effects are listed first. It is assumed that the main effect IDs used start at 1 and go up tp the number of main effects, p.
contrasts	An optional list provided when using a formula. similar to contrasts from glm. See the contrasts.arg of model.matrix.default.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
method	A character string indicating desired method to be used to perform interaction selection. Value can either be "ACTS", where an active set approach is taken and interaction terms are considered for selection only after main effects are brought in, or "HiLa", where the hierarchical lasso penalty is used to ensure hierarchy is maintained in each step. Default Value is "ACTS".
x	Design matrix of dimension length(y) x nvars where each row is represents an obersvation of predictor variables. Assumed to be scaled.
family	Modeling family that describes the marginal distribution of the response. Assumed to be an object such as 'gaussian()' or 'poisson()'
corstr	A character string indicating the desired working correlation structure. The following are implemented : "independence" (default value), "exchangeable", and "ar1".
alpha	An intial guess for the correlation parameter value between -1 and 1 . If left NULL (the default), the initial estimate is 0.
intercept	Binary value indicating where an intercept term is to be included in the model for estimation. Default is to include an intercept.
offset	Vector of offset value(s) for the linear predictor. 'offset' is assumed to be either of length one, or of the same length as 'y'. Default is to have no offset.
control	A list of parameters used to contorl the path generation process; see sgee.control.
standardize	A logical parameter that indicates whether or not the covariates need to be standardized before fitting (but after generating interaction terms from main covariates). If standardized before fitting, the unstandardized path is returned as the default, with a standardizedPath and standardizedX included separately. Default value is TRUE.
verbose	Logical parameter indicating whether output should be produced while isee is running. Default value is FALSE.

Value

Object of class 'sgee' containing the path of coefficient estimates, the path of scale estimates, the path of correlation parameter estimates, and the iteration at which iSEE terminated, and initial regression values including `x`, `y`, codefamily, `clusterID`, `interactionID`, `offset`, `epsilon`, and `numIt`.

Note

While the two different possible methods that can be used with `isee` reflect two different "styles" of stagewise estimation, both achieve a desired hierarchy in the resulting model paths.

When considering models with interaction terms, there are three forms of hierarchy that may be present. Strong hierarchy implies that interaction effects are included in the model only if both of its corresponding main effects are also included in the model. Weak hierarchy implies that an interaction effect can be in the model only if AT LEAST one of its corresponding main effects is also included. The third type of hierarchy is simply a lack of hierarchy; that is an interaction term can be included regardless of main effects.

In practice strong hierarchy is usually what is desired as it is the simplest to interpret, but requires a higher amount of computation when performing model selection. Weak hierarchy is sometimes used as a compromise between the interpret-ability of strong hierarchy and the computational ease of no hierarchy. Both `isee` methods only implement strong hierarchy as the use of stagewise procedures greatly reduces the computational burden.

The active set appraoch, ACTS, tends to have slightly better predictive and model selection performance when the true model is closer to a purely strong hierarchy, but HiLa tends to do better if the true model hierarchy is closer to having a purely weak hierarchy. Thus, in practice, it is important to use external information and judgement to determine which approach is more appropriate.

Author(s)

Gregory Vaughan

References

Vaughan, G., Aseltine, R., Chen, K., Yan, J., (2017). Efficient interaction selection for clustered data via stagewise generalized estimating equations. Department of Statistics, University of Connecticut. Technical Report.

Zhu, R., Zhao, H., and Ma, S. (2014). Identifying gene-environment and gene-gene interactions using a progressive penalization approach. Genetic Epidemiology 38, 353–368.

Bien, J., Taylor, J., and Tibshirani, R. (2013). A lasso for hierarchical interactions. The Annals of Statistics 41, 1111–1141.

Examples

```
#####################
## Generate test data
#####################

## Initialize covariate values
```

```
p <- 5
beta <- c(1, 0, 1.5, 0, .5, ## Main effects
          rep(0.5,4), ## Interaction terms
          0.5, 0, 0.5,
          0,1,
          0)


generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         beta = beta,
                         numMainEffects = p,
                         family = gaussian(),
                         intercept = 1)


## Perform Fitting by providing formula and data
genDF <- data.frame(Y = generatedData$y, X = generatedData$xMainEff)

## Using "ACTS" method
coefMat1 <- isee(formula(paste0("Y~(",
                                paste0("X.", 1:p, collapse = "+"),
                                  ")^2")),
                 data = genDF,
                 family = gaussian(),
                 clusterID = generatedData$clusterID,
                 corstr = "exchangeable",
                 method = "ACTS",
                 control = sgee.control(maxIt = 50, epsilon = 0.5))

## Using "HiLa" method
coefMat2 <- isee(formula(paste0("Y~(",
                                paste0("X.", 1:p, collapse = "+"),
                                  ")^2")),
                 data = genDF,
                 family = gaussian(),
                 clusterID = generatedData$clusterID,
                 corstr = "exchangeable",
                 method = "HiLa",
                 control = sgee.control(maxIt = 50, epsilon = 0.5))
```

---

plot.sgee *Coefficient Traceplot Function*

Description

Function to produce the coefficent traceplot, with capabilities to account for covariate groups. Used in place of the `plot` function.

Usage

```
## S3 method for class 'sgee'
plot(x, y, penaltyFun = NULL, main = NULL,
  xlab = "Iterations", ylab = expression(beta), dropIntercept = FALSE,
  trueBeta = NULL, color = TRUE, manualLineColors = NULL,
  pointSpacing = 3, cutOff = NULL, ...)

## S3 method for class 'sgeeSummary'
plot(x, y, ...)
```

Arguments

`x`	Path of coefficient Estimates.
`y`	Optional parameter inherited from `plot(x,y,...)`; not used with sgee.
`penaltyFun`	Optional function that when provided results ina plot of the coefficient estimates verus the corresponding penalty value. When no `penaltyFun` value is given, the plot generated is of the coefficent estimates versus the iteration number.
`main`	Optional title of plot.
`xlab`	Label of x axis; default value is 'Iterations'.
`ylab`	Label of y axis; default value is the beta symbol.
`dropIntercept`	Logical parameter indicating whether the intercept estimates should be dropped from the plot (i.e. not plotted). The default is FALSE.
`trueBeta`	The true coefficient values. If the true coefficient values can be provided, then coefficient estimates that are false positive identifications as non-zero are marked in the plot.
`color`	Logical parameter indicating that a plot using colors to differentiate coefficients is desired.
`manualLineColors`	Vector of desired line colors; must match dimension of line colors needed (i.e. same number of colors as there are groups if grouped covariates are sharing a color).
`pointSpacing`	Space between marks used to indicate a coefficient is a false positive. Spacing is measured in terms of number of indices of the path matrix between marks.
`cutOff`	Integer value indicating that only the first `cutOff` steps are to be plotted. Default value is `NULL`, indicating all steps are to be plotted.
`...`	Not currently used.

Details

plot.sgee is meant to allow for easy visualization of paths of stagewise (or regularized) coefficient estimates. A great deal of flexibility is provided in terms of how the plot is presented. The poenaltyFun paramter allows for a penalty function to be provided (such as the L_1 norm) to plot the coefficietn estimates against. When given the trueBeta parameter, the plot marks the paths of coefficient estimates that are falsely identified as being non zero. Finally, a switch for black and white versus color plots is provided (`color`).

Note

Function is intended to give a visual representation of the coefficient estimates. Which x values to compare the estimates to can depend on the situation, but typically the most versatile measure to use is the sum of absolute values, the L_1 norm; especially when comparing different coefficient paths from different techniques.

Author(s)

Gregory Vaughan

Examples

```
#####################
## Generate test data
#####################

## Initialize covariate values
p <- 50
beta <- c(rep(2.4,5),
          c(1.3, 0, 1.7, 0, .5),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 1
numGroups <- length(beta)/groupSize



generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         numGroups = numGroups,
                         groupSize = groupSize,
                         groupRho = 0.3,
                         beta = beta,
                         family = gaussian(),
                         intercept = 0)
```

```
genDF <- data.frame(generatedData$y, generatedData$x)
coefMat <- bisee(formula(genDF),
                 data = genDF,
                 lambda1 = 0,         ##effectively see
                 lambda2 = 1,
                 family = gaussian(),
                 clusterID = generatedData$clusterID,
                 corstr="exchangeable",
                 maxIt = 200,
                 epsilon = .1)
############################
## Various options for plots
############################

par(mfrow = c(2,2))

## plain useage
plot(coefMat, main = "Plain Usage")

## With penalty
plot(coefMat, penaltyFun = function(x){sum(abs(x))}, xlab
= expression(abs(abs(beta))[1]), main = "With Penalty")

## using true beta value to highlight misclassifications
plot(coefMat, trueBeta = beta, main = "ID Missclassification")

## black and white option
plot(coefMat, trueBeta = beta, color = FALSE, main =
"Black and White", pointSpacing = 5)
```

`print.sgee` `print` *function for sgee*

Description

Provides implementation of `print` function for `sgee` objects.

Usage

```
## S3 method for class 'sgee'
print(x, ...)
```

Arguments

`x`	Object of class `sgee`, from which various path information is pulled.
`...`	Not currently used

Author(s)

Gregory Vaughan

Examples

```
#####################
## Generate test data
#####################

## Initialize covariate values
p <- 50
beta <- c(rep(2,5),
          c(1, 0, 1.5, 0, .5),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 5
numGroups <- length(beta)/groupSize


generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         numGroups = numGroups,
                         groupSize = groupSize,
                         groupRho = 0.3,
                         beta = beta,
                         family = gaussian(),
                         intercept = 1)

genDF <- data.frame(generatedData$y, generatedData$x)
names(genDF) <- c("Y", paste0("Cov", 1:p))
coefMat <- hisee(formula(genDF), data = genDF,
                 family = gaussian(),
                 clusterID = generatedData$clusterID,
                 groupID = generatedData$groupID,
                 corstr="exchangeable",
                 maxIt = 50,
                 epsilon = .5)

print(coefMat)
```

---

print.sgeeSummary print *function for sgee summaries*

Description

Provides implementation of `print` function for summaries of sgee objects.

Usage

```
## S3 method for class 'sgeeSummary'
print(x, ...)
```

Arguments

`x`	An object of the `sgeeSummary` class, produced by applying the `summary` function to an object of class `sgee`.
`...`	Not currently used

Author(s)

Gregory Vaughan

Examples

```
####################
## Generate test data
####################

## Initialize covariate values
p <- 50
beta <- c(rep(2,5),
          c(1, 0, 1.5, 0, .5),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 5
numGroups <- length(beta)/groupSize


generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         numGroups = numGroups,
                         groupSize = groupSize,
                         groupRho = 0.3,
                         beta = beta,
                         family = gaussian(),
                         intercept = 1)

genDF <- data.frame(generatedData$y, generatedData$x)
names(genDF) <- c("Y", paste0("Cov", 1:p))
coefMat <- hisee(formula(genDF), data = genDF,
```

```
                  family = gaussian(),
                  clusterID = generatedData$clusterID,
                  groupID = generatedData$groupID,
                  corstr="exchangeable",
                  maxIt = 50,
                  epsilon = .5)

sgeeSum <- summary(coefMat)
print(sgeeSum)
```

---

see *Stagewise Estimating Equations Implementation*

Description

Function to perform SEE, a Forward Stagewise regression approach for model selection / dimension reduction using Generalized Estimating Equations

Usage

```
see(y, ...)

## S3 method for class 'formula'
see(formula, data = list(), clusterID, waves = NULL,
  contrasts = NULL, subset, ...)

## Default S3 method:
see(y, x, waves = NULL, ...)

## S3 method for class 'fit'
see(y, x, family, clusterID, waves = NULL,
  corstr = "independence", alpha = NULL, intercept = TRUE, offset = 0,
  control = sgee.control(maxIt = 200, epsilon = 0.05, stoppingThreshold =
  min(length(y), ncol(x)) - intercept, undoThreshold = 0), standardize = TRUE,
  verbose = FALSE, ...)
```

Arguments

y	Vector of response measures that corresponds with modeling family given in 'family' parameter. y is assumed to be the same length as clusterID and is assumed to be organized into clusters as dictated by clusterID.
...	Not currently used
formula	Object of class 'formula'; a symbolic description of the model to be fitted
data	Optional data frame containing the variables in the model.

clusterID	Vector of integers that identifies the clusters of response measures in y. Data and clusterID are assumed to 1) be of equal lengths, 2) sorted so that observations of a cluster are in contiguous rows, and 3) organized so that clusterID is a vector of consecutive integers.
waves	An integer vector which identifies components in clusters. The length of waves should be the same as the number of observations. waves is automatically generated if none is supplied, but when using subset parameter, the waves parameter must be provided by the user for proper calculation.
contrasts	An optional list provided when using a formula. similar to contrasts from glm. See the contrasts.arg of model.matrix.default.
subset	An optional vector specifying a subset of observations to be used in the fitting process.
x	Design matrix of dimension length(y) x nvar, the number of variables, where each row is represents an observation of predictor variables.
family	Modeling family that describes the marginal distribution of the response. Assumed to be an object such as gaussian() or poisson().
corstr	A character string indicating the desired working correlation structure. The following are implemented : "independence" (default value), "exchangeable", and "ar1".
alpha	An initial guess for the correlation parameter value between -1 and 1 . If left NULL (the default), the initial estimate is 0.
intercept	Binary value indicating where an intercept term is to be included in the model for estimation. Default is to include an intercept.
offset	Vector of offset value(s) for the linear predictor. offset is assumed to be either of length one, or of the same length as y. Default is to have no offset.
control	A list of parameters used to contorl the path generation process; see sgee.control.
standardize	A logical parameter that indicates whether or not the covariates need to be standardized before fitting. If standardized before fitting, the unstandardized path is returned as the default, with a standardizedPath and standardizedX included separately. Default value is TRUE.
verbose	Logical parameter indicating whether output should be produced while bisee is running. Default value is FALSE.

Details

Function to implement SEE, a stagewise regression approach that is designed to perform model selection in the context of Generalized Estimating Equations. Given a response y and a design matrix x (excluding intercept) SEE generates a path of stagewise regression estimates for each covariate based on the provided step size epsilon.

The resulting path can then be analyzed to determine an optimal model along the path of coefficient estimates. The summary.sgee function provides such functionality based on various possible metrics, primarily focused on the Mean Squared Error. Furthermore, the plot.sgee function can be used to examine the path of coefficient estimates versus the iteration number, or some desired penalty.

A stochastic version of this function can also be called. using the auxiliary function `sgee.control` the parameters `stochastic`, `reSample`, and `withReplacement` can be given to `see` to perform a sub sampling step in the procedure to make the SEE implementation scalable for large data sets.

Value

Object of class `sgee` containing the path of coefficient estimates, the path of scale estimates, the path of correlation parameter estimates, the iteration at which SEE terminated, and initial regression values including `x`, `y`, codefamily, `clusterID`, `groupID`, `offset`, `epsilon`, and `numIt`.

Author(s)

Gregory Vaughan

References

Vaughan, G., Aseltine, R., Chen, K., Yan, J., (2017). Stagewise Generalized Estimating Equations with Grouped Variables. Biometrics 73, 1332-1342. URL: http://dx.doi.org/10.1111/biom.12669, doi:10.1111/biom.12669.

Wolfson, J. (2011). EEBoost: A general method for prediction and variable selection based on estimating equations. Journal of the American Statistical Association 106, 296–305.

Tibshirani, R. J. (2015). A general framework for fast stagewise algorithms. Journal of Machine Learning Research 16, 2543–2588.

Examples

```
#####################
## Generate test data
#####################

## Initialize covariate values
p <- 50
beta <- c(rep(2,5),
          c(1, 0, 1.5, 0, .5),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 1
numGroups <- length(beta)/groupSize


generatedData <- genData(numClusters = 50,
                         clusterSize = 4,
                         clusterRho = 0.6,
                         clusterCorstr = "exchangeable",
                         yVariance = 1,
                         xVariance = 1,
                         numGroups = numGroups,
                         groupSize = groupSize,
                         groupRho = 0.3,
```

```
                        beta = beta,
                        family = gaussian(),
                        intercept = 1)



## Perform Fitting by providing formula and data
genDF <- data.frame(generatedData$y, generatedData$x)
names(genDF) <- c("Y", paste0("Cov", 1:p))
coefMat1 <- see(formula(genDF), data = genDF,
                family = gaussian(),
                waves = rep(1:4, 50),
                clusterID = generatedData$clusterID,
                groupID = generatedData$groupID,
                corstr = "exchangeable",
                control = sgee.control(maxIt = 50, epsilon = 0.5),
                verbose = TRUE)

## set parameter 'stochastic' to 0.5 to implement the stochastic
## stagewise approach where a subsmaple of 50% of the data is taken
## before the path is calculation.
## See sgee.control for more details about the parameters for the
## stochastic stagewise approach

coefMat2 <- see(formula(genDF), data = genDF,
                family = gaussian(),
                waves = rep(1:4, 50),
                clusterID = generatedData$clusterID,
                groupID = generatedData$groupID,
                corstr = "exchangeable",
                control = sgee.control(maxIt = 50, epsilon = 0.5,
                                       stochastic = 0.5),
                verbose = FALSE)

par(mfrow = c(2,1))
plot(coefMat1)
plot(coefMat2)
```

`sgee.control` *Auxiliary for Controlling SGEE fitting*

Description

Auxiliary function for sgee fitting functions. Specifies parameters used by all sgee fitting functions in terms of the path generation; i.e. step size `epsilon`, maximum number of iterations `maxIt`, and the threshold for premature stopping `stoppingthreshold`.

Usage

```
sgee.control(maxIt = 200, epsilon = 0.05, stoppingThreshold = NULL,
  undoThreshold = 0.005, interceptLimit = NULL, stochastic = 1,
  sampleProb = NULL, reSample = 1, withReplacement = FALSE)
```

Arguments

maxIt	Maximum number of iterations of the stagewise algorithm to be executed. Default is 200.
epsilon	Step size to be used when incrementing coefficient value(s) in each iteration. Default is 0.05.
stoppingThreshold	An integer value that indicates the maximum number of allowed covariates in the model. Once the algorithm has reached the value of stoppingThreshold, the algorithm will stop without completing any remaining iterations. The number of covariates to be included cannot exceed the number of observations. The default value is typically the minimum of the number of covariates and the number of observations, minus 1 if an intercept is included.
undoThreshold	A small value used to determine if consecutive steps are sufficiently different. If consecutive steps effectively undo each other (as indicated by having a sum with an absolute value less than undoThreshold), then the steps are repeated and the stepsize is reduced. A negative value for undoThreshold effectively prevents this step. undoThreshold should only be big enough to allow for some rounding error in steps and should be much smaller than the step size. Default value is 0.005.
interceptLimit	sgee functions make use of the extendInt parameter of uniroot to estimate the intercept in each iteration. This parameter was recently implemented and thus may cause issues with older versions of R. If a value is given for interceptLimit, then this extendInt parameter is bypassed and a solution for the intercept estimating equation is sought out between negative interceptLimit and positive interceptLimit. The default value of NULL uses the extendInt functionality.
stochastic	A numeric value between 0 (exclusive) and 1 (inclusive) to indicate what proportion of the data should be subsampled in the stochastic implementation of stagewise approached. The default value of 1 implements the standard deterministic approach where no subsampling is done.
sampleProb	A user provided value dictating the probability distribution for stochastic stagewise approaches. sampleProb can be provided as 1) a vector of fixed values of length equal to the resposne vector y, 2) a function that takes in a list of values (full list of values given in details) and returns a vector of length equal to the response vector y, or 3) the default value of NULL, which results in a uniform distribution
reSample	Parameter indicating how frequently a subsample is collected in stochastic stagewise approaches. If reSample == 1 then a subsample is collected every iteration,if reSample == 2 a subsample is collected every two (i.e every other) iteration. If reSample == 0, (the default value) then a subsample is only collected once before any iterations have been done.

withReplacement
: a Logical value indicating if the subsampling in stochastic stagewise approaches should be done with or without replacement. Default values is FALSE.

Value

A list containing all of the parameter values.

Author(s)

Gregory Vaughan

---

summary.sgee *Coefficient Path summary*

Description

Function to analyze and summarize a path of coefficent values by comparing them using prediction error on a \"new\" data set (or fold in CV), or the original data set if no comparison data is provided. The best point along the path in terms of the prediction error is identified. All of the prediction errors for each point along the path, the minimum prediction error, and the index of the minimum are returned.

Usage

```
## S3 method for class 'sgee'
summary(object, newX = NULL, newY = NULL, newOffset = NULL,
  trueBeta = NULL, trueIntercept = NULL, scale = NULL,
  classification = 0.5, averaged = TRUE, ...)
```

Arguments

object	Object of class sgee, from which various path information is pulled.
newX	Design matrix to be used for model testing. It is assumed that newX does not contain an intercept column. An intercept column is appended bysgee.summary if an intercept was used to make object.
newY	Response vector to be used for model testing.
newOffset	Vector of offsets to be used for model testing. Must be same length as newY.
trueBeta	For simulation use; true coefficient values can be provided to get certain metrics.
trueIntercept	For simulation use; true intercept value to be used in conjunction with trueBeta.
scale	Scale value can be passed to allow for standardized error measurements (poisson case only).
classification	A numeric parameter from 0 to 1 indicating cutoff to be used to determine classification rate in Binomial setting. Default is 0.5. Values below 0 indicate that the squared error, in either the observation or the true linear predictor is the trueBeta is given, is to be used instead of the classification rate.

averaged	Logical parameter indicating whether the mean of the total error is to be used; assumed TRUE.
...	Currently not used.

Details

The prediction error used is dependent on the input. If the true Beta is not given, then the sum squared error (or MSE; see parameter averaged) in the response is used for gaussian (or non-poisson); for poisson if the scale (or an estimate) is also given, then the sum squared Pearson residuals are used, otherwise the deviance is used. If the true Beta is provided then the sum squared error in the linear predictor is used instead.

Furthermore, when true Beta is supplied, additional model selection metrics are produced, including: False Positive Rate, False Discovery Rate, False Negative Rate.

The function is provided to allow for model selection; given a path generated by a sgee function, the path can be fed into this function with a testing data set to identify an optimal point along the path. Cross validation can be performed by dividing the original data set into k folds before hand and generating multiple coefficient paths and applying this function to each path generated.

Value

A list containing 1) a vector of prediction errors with testing data set, 2) the smallest prediction error found along path, 3) the index of the smallest error, and if the trueBeta parameter is provided the False Positive, False Discovery, and false negative rates, and True positive and False Positive counts at the index of the smallest error, along with the minimum mis-classification and corresponding index, where the mis-classification is the total of the coefficients incorrectly marked as important/unimportant.

Author(s)

Gregory Vaughan

Examples

```
## Initialize covariate values
p <- 50
beta <- c(rep(2.4,5),
          c(1.3, 0, 1.7, 0, .5),
          rep(0.5,5),
          rep(0,p-15))
groupSize <- 1
numGroups <- length(beta)/groupSize



trainingData <- genData(numClusters = 50,
                        clusterSize = 4,
                        clusterRho = 0.6,
                        clusterCorstr = "exchangeable",
                        yVariance = 1,
```

```
                          xVariance = 1,
                          numGroups = numGroups,
                          groupSize = groupSize,
                          groupRho = 0.3,
                          beta = beta,
                          family = gaussian(),
                          intercept = 1)

testingData <- genData(numClusters = 50,
                       clusterSize = 4,
                       clusterRho = 0.6,
                       clusterCorstr = "exchangeable",
                       yVariance = 1,
                       xVariance = 1,
                       numGroups = numGroups,
                       groupSize = groupSize,
                       groupRho = 0.3,
                       beta = beta,
                       family = gaussian(),
                       intercept = 1)

coefMat <- see(y = trainingData$y,
               x = trainingData$x,
                    family = gaussian(),
                    clusterID = trainingData$clusterID,
                    corstr="exchangeable",
                    maxIt = 200,
                    epsilon = .1)

analysisResults <- summary(coefMat,
                           newX = testingData$x,
                           newY = testingData$y)
```

Index