pamr

November 11, 2009

R topics documented:

khan *Khan microarray data*

Description

The khan data frame has 2309 rows and 65 columns. These are one of the datasets data used in the Tibshirani et al paper in PNAS on nearest shrunken centroids.

Details

The first row contains the sample labels. The first two columns of gene ids and names. The remaining values of the matrix are gene expression values.

pamr.adaptthresh *A function to adaptive choose threshold scales, for use in pamr.train*

Description

A function to adaptive choose threshold scales, for use in pamr.train

Usage

```
pamr.adaptthresh(object, ntries = 10, reduction.factor = 0.9, full.out = FALSE)
```

Arguments

object	The result of a call to pamr.train
ntries	Number of iterations to use in algorithm
reduction.factor	Amount by which a scaling is reduced in one step of the algorithm
full.out	Should full output be returned? Default FALSE

Details

pamr.adaptthresh Adaptively searches for set of good threshold scales. The baseline (default) scale is 1 for each class. The idea is that for easy to classify classes, the threshold scale can be increased without increasing the error rate for that class, and resulting in fewer genes needed for the classification rule. The scalings from pamr.adaptthresh are then used in pamr.train, and pamr.cv. The results may be better than those obtained with the default values of threshold.scale.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

References

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. "Diagnosis of multiple cancer types by shrunken centroids of gene expression" PNAS 2002 99:6567-6572 (May 14).

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu (2002). Class prediction by nearest shrunken centroids,with applications to DNA microarrays. Stanford tech report.

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
new.scales <- pamr.adaptthresh(mytrain)


mytrain2 <- pamr.train(mydata, threshold.scale=new.scales)

myresults2 <- pamr.cv(mytrain2, mydata)
```

---

pamr.batchadjust *A function to mean-adjust microarray data by batches*

Description

A function to mean-adjust microarray data by batches

Usage

```
pamr.batchadjust(data)
```

Arguments

data	The input data. A list with components: x- an expression genes in the rows, samples in the columns, and y- a vector of the class labels for each sample, and batchlabels- a vector of batch labels for each sample.

This object if the same form as that produced by pamr.from.excel.

Details

pamr.batchadjust does a genewise one-way ANOVA adjustment for expression values. Let x(i,j) be the expression for gene i in sample j. Suppose sample j in in batch b, and let B be the set of all samples in batch b. Then pamr.batchadjust adjusts x(i,j) to x(i,j) - mean[x(i,j)] where the mean is taken over all samples j in B

Value

A data object of the same form as the input data, with x replaced by the adjusted x

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
#generate some data
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
batchlabels <- sample(c(1:5),size=20,replace=TRUE)
mydata <- list(x=x,y=factor(y),batchlabels=factor(batchlabels))

mydata2 <- pamr.batchadjust(mydata)
```

---

pamr.confusion *A function giving a table of true versus predicted values, from a nearest shrunken centroid fit.*

Description

A function giving a table of true versus predicted values, from a nearest shrunken centroid fit.

Usage

```
pamr.confusion(fit, threshold, extra=TRUE)
```

Arguments

fit	The result of a call to pamr.train or pamr.cv
threshold	The desired threshold value
extra	Should the classwise and overall error rates be returned? Default TRUE

Details

pamr.confusion Gives a cross-tabulation of true versus predicted classes for the fit returned by pamr.train or pamr.cv, at the specified threshold.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
mycv <- pamr.cv(mytrain,mydata)
pamr.confusion(mytrain,  threshold=2)
pamr.confusion(mycv,  threshold=2)
```

`pamr.confusion.survival` *Compute confusin matrix from pamr survival fit*

Description

computes confusion matrix for (survival.time,censoring) outcome based on fit object "fit" and class predictions "yhat" soft response probabilities for (survival.time,censoring) are first estimated using Kaplan-Meier method applied to training data

Usage

```
pamr.confusion.survival(fit, survival.time, censoring.status, yhat)
```

Arguments

`fit`	The result of a call to pamr.train or pamr.cv
`survival.time`	Survival time
`censoring.status`	censoring status
`yhat`	class predictions

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

`pamr.cv` *A function to cross-validate the nearest shrunken centroid classifier*

Description

A function to cross-validate the nearest shrunken centroid classifier produced by pamr.train

Usage

```
pamr.cv(fit, data,  nfold = NULL, folds = NULL,...)
```

Arguments

`fit`	The result of a call to pamr.train
`data`	A list with at least two components: x- an expression genes in the rows, samples in the columns), and y- a vector of the class labels for each sample. Same form as data object used by pamr.train.
`nfold`	Number of cross-validation folds. Default is the smallest class size
`folds`	A list with nfold components, each component a vector of indices of the samples in that fold. By default a (random) balanced cross-validation is used
`...`	Any additional arguments that are to be passed to pamr.train

Details

`pamr.cv` carries out cross-validation for a nearest shrunken centroid classifier.

Value

A list with components

`threshold`	A vector of the thresholds tried in the shrinkage
`errors`	The number of cross-validation errors for each threshold value
`loglik`	The cross-validated multinomial log-likelihood value for each threshold value
`size`	A vector of the number of genes that survived the thresholding, for each threshold value tried.
`yhat`	A matrix of size n by nthreshold, containing the cross-validated class predictions for each threshold value, in each column
`prob`	A matrix of size n by nthreshold, containing the cross-validated class probabilities for each threshold value, in each column
`folds`	The cross-validation folds used
`cv.objects`	Train objects (output of pamr.train), from each of the CV folds
`call`	The calling sequence used

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)

mydata <- list(x=x, y=factor(y), geneid=as.character(1:nrow(x)),
               genenames=paste("g", as.character(1:nrow(x)), sep=""))

mytrain <-   pamr.train(mydata)
mycv <- pamr.cv(mytrain, mydata)
```

---

pamr.decorrelate *A function to decorrelate (adjust) the feature matrix with respect to some additional predictors*

Description

A function to decorrelate (adjust) the feature matrix with respect to some additional predictors

Usage

```
pamr.decorrelate(x, adjusting.predictors, xtest=NULL, adjusting.predictors.test
```

Arguments

x	Matrix of training set feature values, with genes in the rows, samples in the columns
adjusting.predictors	List of training set predictors to be used for adjustment
xtest	Optional matrix of test set feature values, to be adjusted in the same way as the training set
adjusting.predictors.test	Optional list of test set predictors to be used for adjustment

Details

pamr.decorrelate Does a least squares regression of each row of x on the adjusting predictors, and returns the residuals. If xtest is provided, it also returns the adjusted version of xtest, using the training set least squares regression model for adjustment

Value

A list with components

x.adj	Adjusted x matrix
xtest.adj	Adjusted xtest matrix, if xtest we provided

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

References

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu Diagnosis of multiple cancer types by shrunken centroids of gene expression PNAS 99: 6567-6572. Available at www.pnas.org

Examples

```
#generate some data
set.seed(120)

x<-matrix(rnorm(1000*20),ncol=20)
y<-c(rep(1,10),rep(2,10))
adjusting.predictors=list(pred1=rnorm(20), pred2=as.factor(sample(c(1,2),replace
=TRUE,size=20)))
xtest=matrix(rnorm(1000*10),ncol=10)
adjusting.predictors.test=list(pred1=rnorm(10), pred2=as.factor(sample(c(1,2),replace
=TRUE,size=10)))

# decorrelate training x wrt adjusting predictors

x.adj=pamr.decorrelate(x,adjusting.predictors)$x.adj
# train classifier with adjusted x

d=list(x=x.adj,y=y)
a<-pamr.train(d)
```

```
# decorrelate training and test x wrt adjusting predictors, then make
#predictions for test set

temp<-pamr.decorrelate(x,adjusting.predictors, xtest=xtest, adjusting.predictors.test=adj

d=list(x=temp$x.adj,y=y)
a<-pamr.train(d)
aa<-pamr.predict(a,temp$xtest.adj, threshold=.5)
```

`pamr.fdr` *A function to estimate false discovery rates for the nearest shrunken centroid classifier*

Description

A function to estimate false discovery rates for the nearest shrunken centroid classifier

Usage

```
pamr.fdr(trained.obj, data,  nperms=100,
 xl.mode=c("regular","firsttime","onetime","lasttime"),xl.time=NULL, xl.prevfit=
```

Arguments

`trained.obj`	The result of a call to pamr.train
`data`	Data object; same as the one passed to pamr.train
`nperms`	Number of permutations for estimation of FDRs. Default is 100
`xl.mode`	Used by Excel interface
`xl.time`	Used by Excel interface
`xl.prevfit`	Used by Excel interface

Details

`pamr.fdr` estimates false discovery rates for a nearest shrunken centroid classifier

Value

A list with components:

`results`	Matrix of estimates FDRs for various various threshold values. Reported are both the median and 90th percentile of the FDR over permutations
`pi0`	The estimated proportion of genes that are null, i.e. not significantly different

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)

mydata <- list(x=x,y=factor(y), geneid=as.character(1:nrow(x)),
               genenames=paste("g", as.character(1:nrow(x)), sep=""))

mytrain <-   pamr.train(mydata)
myfdr <- pamr.fdr(mytrain, mydata)
```

---

pamr.from.excel *A function to read in a text file saved from Excel*

Description

A function to read in a text file saved from Excel. The spreadsheet is assumed to be of the format used by the SAM program.

Usage

```
pamr.from.excel(file, ncols, sample.labels = FALSE, batch.labels = FALSE)
```

Arguments

file	Character name of a text file. This is assumed to be a tab-delimited text file saved from an excel spreadsheet from "SAM". The spreadsheet has one row of expression values per gene. In addition there is one information row and two information columns. The first row has class labels for each of the samples. The first column had gene identifiers, and the second column has gene names. In the SAM program, for the multiclass option, the samples must be labelled 1,2,3 etc. Here we allow general labels, like "lymphoma", "colon cancer" etc
ncols	Number of columns in file
sample.labels	Optional argument. If true, "file" is assumed to have an additional row at the top, consisting of two blank cells followed by a sample labels for each of the columns. If available, these sample labels are used by various plotting routines.
batch.labels	Optional argument. If true, "file" is assumed to have an additional row at the top, consisting of two blank cells followed by a batch labels for each of the columns. If sample.labels=T as well, the row of batch labels are assumed to come after the row of sample labels. The batch labels are used by the function pamr.batchadjust.

Details

pamr.from.excel Reads in the text file "file", and creates an object with components x (the matrix of expression values), y- a vector of class labels for each sample, geneid- a vector of gene identifiers and genenames- a vector of gene names

Value

A list with components

x	the matrix of expression values
y	a vector of class labels for each sample,
geneid	a vector of gene identifiers
genenames	a vector of gene names
samplelabels	a vector of sample labels, if provided in "file"
batchlabels	a vector of batch labels, if provided in "file"

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

---

pamr.geneplot *A function to plot the genes that surive the thresholding from the nearest shrunken centroid classifier*

Description

A function to plot the genes that survive the thresholding, from the nearest shrunken centroid classifier produced by pamr.train

Usage

```
pamr.geneplot(fit, data, threshold)
```

Arguments

fit	The result of a call to pamr.train
data	The input data. In the same format as the input data for pamr.train
threshold	The desired threshold value

Details

pamr.geneplot Plots the raw gene expression for genes that survive the specified threshold. Plot is stratified by class. Plot is set up to display only up to about 20 or 25 genes, otherwise it gets too crowded. Hence threshold should be chosen to yield at most about 20 or 25 genes.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
pamr.geneplot(mytrain, mydata, threshold=1.6)
```

pamr.indeterminate *A function that takes estimate class probabilities and produces a class prediction or indeterminate prediction*

Description

A function that takes estimate class probabilities and produces a class prediction or indeterminate prediction

Usage

```
pamr.indeterminate(prob, mingap=0)
```

Arguments

prob	Estimated class probabilities, from pamr.predict with type="posterior")
mingap	Minimum difference between highest and second highest probability. If difference is < mingap, prediction is set to indeterminate (NA)

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
prob<- pamr.predict(mytrain, mydata$x , threshold=1, type="posterior")
pamr.indeterminate(prob,mingap=.75)
```

pamr-internal *Internal pamr functions*

Description

Internal pamr functions

Usage

```
pamr.pairscore(x, pair.ind=NULL)
pamr.pvalue.survival(group, survival.time, censoring.status, ngroup.survival)
pamr.score.to.class1(x, scores, cutoff=2, n.class=2)
pamr.score.to.class2(x, scores, cutoff=2, n.pc=1, n.class=2)
pamr.knnimpute.old(data, k = 10)
pamr.cube.root(x)
print.nsc(x, ...)
print.nsccv(x, ...)
```

```
print.pamrtrained(x, ...)
print.pamrcved(x, ...)
pamr.xl.error.trace()
pamr.xl.get.threshold.range(fit)
pamr.xl.get.soft.class.labels(fit, survival.times, censoring.status)
pamr.xl.compute.offset(data, offset.percent=50, prior=prior)
pamr.xl.get.offset()
pamr.xl.derive.adjusted.prior(prior, data)
pamr.xl.get.default.training.parameters(data)
pamr.xl.get.uniform.prior(data, nclasses=NULL)
pamr.xl.get.sample.prior(data)
pamr.xl.get.class.names()
pamr.xl.get.class.labels()
pamr.xl.get.number.of.classes()
pamr.xl.process.data(use.old.version=FALSE)
pamr.xl.compute.cv.confusion (fit, cv.results, threshold)
pamr.xl.compute.confusion (fit, threshold)
pamr.xl.is.a.subset(a, y)
pamr.xl.listgenes.compute (fit, data, threshold, fitcv=NULL,  genenames = FALSE)
pamr.xl.plot.test.probs.compute(fit, new.x, newx.classes, missing.class.label,th
pamr.xl.plot.training.error.compute(trained.object)
pamr.xl.plotcen.compute(fit, data, threshold)
pamr.xl.plotcv.compute(aa)
pamr.xl.plotcvprob.compute(fit, data, threshold)
pamr.xl.predict.test.class(fit, newx, threshold, test.class.labels)
pamr.xl.predict.test.surv.class(fit, newx, threshold, survival.times, censoring.
pamr.xl.predict.test.class.only(fit, newx, threshold)
pamr.xl.predict.test.probs(fit, newx, threshold)
pamr.xl.test.data.impute(x, k, use.old.version=FALSE)
pamr.xl.test.errors.surv.compute(fit, newx, threshold=fit$threshold, survival.ti
pamr.xl.test.errors.compute(fit, newx, newx.classes, threshold=fit$threshold, pr
pamr.xl.transform.class.labels(x)
pamr.xl.transform.data(data)
pamr.xl.transform.test.data(test.x)
pamr.xl.plotsurvival(fit, data, threshold)
pamr.xl.plotsurvival.test(fit, newx, survival.time, censoring.status, threshold)
pamr.xl.plotsurvival.strata(fit, data)
pamr.xl.test.get.soft.classes(fit, survival.times, censoring.status)
pamr.xl.get.soft.class.labels(fit, survival.times, censoring.status)
```

Details

These are not to be called by the user.

Author(s)

Balasubramanian Narasimhan and Rob Tibshirani

`pamr.knnimpute` *A function to impute missing expression data*

Description

A function to impute missing expression data, using nearest neighbor averaging.

Usage

```
pamr.knnimpute(data ,k = 10, rowmax = 0.5, colmax = 0.8, maxp = 1500)
```

Arguments

`data`	The PAM input data. A list with components: `x`, an expression matrix with genes in the rows, samples in the columns, and `y`, a vector of the class labels for each sample. Same form as used by pamr.train, and same as that produced by `pamr.from.excel`
`k`	Number of neighbors to be used in the imputation (default=10)
`rowmax`	The maximum percent missing data allowed in any row (default 50%). For any rows with more than `rowmax`% missing are imputed using the overall mean per sample.
`colmax`	The maximum percent missing data allowed in any column (default 80%). If any column has more than `colmax`% missing data, the program halts and reports an error.
`maxp`	The largest block of genes imputed using the knn algorithm inside `pamr.knnimpute` (default 1500); larger blocks are divided by two-means clustering (recursively) prior to imputation. If `maxp=p`, only knn imputation is done.

Details

`pamr.knnimpute` uses k-nearest neighbors in the space of genes to impute missing expression values.

For each gene with missing values, we find the k nearest neighbors using a Euclidean metric, confined to the columns for which that gene is NOT missing. Each candidate neighbor might be missing some of the coordinates used to calculate the distance. In this case we average the distance from the non-missing coordinates. Having found the k nearest neighbors for a gene, we impute the missing elements by averaging those (non-missing) elements of its neighbors. This can fail if ALL the neighbors are missing in a particular element. In this case we use the overall column mean for that block of genes.

Since nearest neighbor imputation costs $O(p\log(p))$ operations per gene, where p is the number of rows, the computational time can be excessive for large p and a large number of missing rows. Our strategy is to break blocks with more than `maxp` genes into two smaller blocks using two-mean clustering. This is done recursively till all blocks have less than `maxp` genes. For each block, knn imputation is done separately. We have set the default value of `maxp` to 1500. Depending on the speed of the machine, and number of samples, this number might be increased. Making it too small is counter-productive, because the number of two-mean clustering algorithms will increase.

Value

`data`	The input data list, with x replaced by the imputed version of x

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

References

Hastie, T., Tibshirani, R., Sherlock, G., Eisen, M., Brown, P. and Botstein, D., Imputing Missing Data for Gene Expression Arrays, Stanford University Statistics Department Technical report (1999), http://www-stat.stanford.edu/~hastie/Papers/missing.pdf

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein and Russ B. Altman, Missing value estimation methods for DNA microarrays BIOINFORMATICS Vol. 17 no. 6, 2001 Pages 520-525

---

pamr.listgenes *A function to list the genes that survive the thresholding, from the nearest shrunken centroid classifier*

Description

A function to list the genes that survive the thresholding, from the nearest shrunken centroid classifier produced by pamr.train.

Usage

```
pamr.listgenes(fit, data,  threshold, fitcv=NULL, genenames=FALSE)
```

Arguments

fit	The result of a call to pamr.train
data	The input data. In the same format as the input data for pamr.train
threshold	The desired threshold value
fitcv	Optional object, result of a call to pamr.cv
genenames	Include genenames in the list? If yes, they are taken from "data". Default is false (geneid is always included in the list).

Details

pamr.listgenes List the geneids, and standardized centroids for each class, for genes surviving at the given threshold. If fitcv is provided, the function also reports the average rank of the gene in the cross-validation folds, and the proportion of times that the gene is chosen (at the given threshold) in the cross-validation folds.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
#generate some data
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)

mydata <- list(x=x,y=factor(y), geneid=as.character(1:nrow(x)),
               genenames=paste("g", as.character(1:nrow(x)), sep=""))

#train classifier
mytrain<-   pamr.train(mydata)

pamr.listgenes(mytrain, mydata, threshold=1.6)
```

`pamr.makeclasses` *A function to interactively define classes from a clustering tree*

Description

function to interactively define classes from a clustering tree

Usage

```
pamr.makeclasses(data,sort.by.class=FALSE,...)
```

Arguments

`data`	The input data. A list with components: x- an expression genes in the rows, samples in the columns, and y- a vector of the class labels for each sample, and batchlabels- a vector of batch labels for each sample. This object if the same form as that produced by pamr.from.excel.
`sort.by.class`	Optional argument. If true, the clustering tree is forced to put all samples in the same class (as defined by the class labels y in 'data') together in the tree. This is useful if a regrouping of classes is desired. Eg: given classes 1,2,3,4 you want to define new classes (1,3) vs (2,4) or 2 vs (1,3)
`...`	Any additional arguments to be passed to hclust

Details

`pamr.makeclasses` Using this function the user interactively defines a new set of classes, to be used in pamr.train, pamr.cv etc. After invoking pamr.makeclasses, a clustering tree is drawn. This callss the R function `hclust`, and any arguments for `hclust` can be passed to it. Using the left button, the user clicks at the junction point defining the subgroup 1. More groups can be added to class 1 by clicking on further junction points. The user ends the definition of class 1 by clicking on the rightmost button [in Windows, an additional menu appears and he chooses Stop] . This process is continued for classes 2,3 etc. Note that some sample may be left out of the new classes. Two consecutive clicks of the right button ends the definition for all classes.

At the end, the clustering is redrawn, with the new class labels shown.

Note: this function is "fragile". The user must click close to the junction point, to avoid confusion with other junction points. Classes 1,2,3.. cannot have samples in common (if they do, an Error message will appear). If the function is confused about the desired choices, it will complain and ask the user to rerun pamr.makeclasses. The user should also check that the labels on the final redrawn cluster tree agrees with the desired classes.

Value

A vector of class labels 1,2,3... If a component is NA (missing), then the sample is not assigned to any class. This vector should be assigned to the newy component of data, for use in pamr.train etc. Note that pamr.train uses the class labels in the component "newy" if it is present. Otherwise it uses the data labels "y".

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
#generate some data
x <- matrix(rnorm(1000*40),ncol=40)
y <- sample(c(1:4),size=40,replace=TRUE)
batchlabels <- sample(c(1:5),size=40,replace=TRUE)

mydata <- list(x=x,y=factor(y),batchlabels=factor(batchlabels),
               geneid=as.character(1:nrow(x)),
               genenames=paste("g",as.character(1:nrow(x)),sep=""))

# mydata$newy <- pamr.makeclasses(mydata)    Run this and define some new classes

train <- pamr.train(mydata)
```

---

pamr.menu *A function that interactively leads the user through a PAM analysis*

Description

A function that interactively leads the user through a PAM analysis

Usage

```
pamr.menu(data)
```

Arguments

data	A list with at least two components: x- an expression genes in the rows, samples in the columns), and y- a vector of the class labels for each sample. Same form as data object used by pamr.train.

Details

pamr.menu provides a menu for training, cross-validating and plotting a nearest shrunken centroid analysis.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
#  pamr.menu(mydata)
```

---

pamr.plotcen *A function to plot the shrunken class centroids, from the nearest shrunken centroid classifier*

Description

A function to plot the shrunken class centroids, from the nearest shrunken centroid classifier produced by pamr.train.

Usage

```
pamr.plotcen(fit, data,  threshold)
```

Arguments

fit	The result of a call to pamr.train
data	The input data, in the same form as that used by pamr.train
threshold	The desired threshold value

Details

pamr.plotcen plots the shrunken class centroids for each class, for genes surviving the threshold for at least once class. If genenames are included in "data", they are added to the plot. Note: for many classes and long gene names, this plot may need some manual prettying.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y,genenames=as.character(1:1000))
mytrain <-   pamr.train(mydata)
mycv <- pamr.cv(mytrain,mydata)
pamr.plotcen(mytrain, mydata,threshold=1.6)
```

`pamr.plotcvprob` *A function to plot the cross-validated sample probabilities from the nearest shrunken centroid classifier*

Description

A function to plot the cross-validated sample probabilities from the nearest shrunken centroid classifier

Usage

```
pamr.plotcvprob(fit, data, threshold)
```

Arguments

`fit`	The result of a call to pamr.cv
`data`	A list with at least two components: x- an expression genes in the rows, samples in the columns), and y- a vector of the class labels for each sample. Same form as data object used by pamr.train.
`threshold`	Threshold value to be used

Details

`pamr.plotcvprob` plots the cross-validated sample probabilities the from nearest shrunken centroid classifier, stratified by the true classses.

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
mycv <-  pamr.cv(mytrain,mydata)
pamr.plotcvprob(mycv,mydata,threshold=1.6)
```

pamr.plotcv — *A function to plot the cross-validated error curves from the nearest shrunken centroid classifier*

Description

A function to plot the cross-validated error curves the nearest shrunken centroid classifier

Usage

```
pamr.plotcv(fit)
```

Arguments

fit	The result of a call to pamr.cv

Details

pamr.plotcv plots the cross-validated misclassification error curves, from nearest shrunken centroid classifier. An overall plot, and a plot by class, are produced.

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
mycv <-  pamr.cv(mytrain, mydata)
pamr.plotcv(mycv)
```

pamr.plotfdr — *A function to plot the FDR curve from the nearest shrunken centroid classifier*

Description

A function to plot the FDR curve the nearest shrunken centroid classifier

Usage

```
pamr.plotfdr(fdrfit,  call.win.metafile = FALSE)
```

Arguments

fdrfit	The result of a call to pamr.fdr
call.win.metafile	Used by Excel interface

Details

`pamr.plotfdr` plots the FDR curves from nearest shrunken centroid classifier. The median FDR (solid line) and upper 90 percentile (broken line) are shown

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:2),size=20,replace=TRUE)
x[1:50,y==2]=x[1:50,y==2]+3
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
myfdr <-  pamr.fdr(mytrain, mydata)
pamr.plotfdr(myfdr)
```

---

pamr.plotstrata *A function to plot the survival curves in each Kaplan Meier stratum*

Description

A function to plot the survival curves in each Kaplan Meier stratum

Usage

```
pamr.plotstrata(fit, survival.time, censoring.status)
```

Arguments

`fit`	The result of a call to pamr.train
`survival.time`	Vector of survival times
`censoring.status`	Vector of censoring status values

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
gendata<-function(n=100, p=2000){
  tim <- 3*abs(rnorm(n))
  u<-runif(n,min(tim),max(tim))
  y<-pmin(tim,u)
   ic<-1*(tim<u)
m <- median(tim)
x<-matrix(rnorm(p*n),ncol=n)
  x[1:100, tim>m] <-  x[1:100, tim>m]+3
```

```
  return(list(x=x,y=y,ic=ic))
}

# generate training data; 2000 genes, 100 samples

junk<-gendata(n=100)
y<-junk$y
ic<-junk$ic
x<-junk$x
d <- list(x=x, survival.time=y, censoring.status=ic,
          geneid=as.character(1:nrow(x)),
          genenames=paste("g", as.character(1:nrow(x)),sep=""))

# train model
a3<- pamr.train(d, ngroup.survival=2)

pamr.plotstrata(a3, d$survival.time, d$censoring.status)
```

`pamr.plotsurvival` *A function to plots Kaplan-Meier curves stratified by a group variable*

Description

A function to plots Kaplan-Meier curves stratified by a group variable

Usage

```
pamr.plotsurvival(group, survival.time, censoring.status)
```

Arguments

`group`	A grouping factor
`survival.time`	Vector of survival times
`censoring.status`	Vector of censoring status values: 1=died, 0=censored

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
gendata<-function(n=100, p=2000){
  tim <- 3*abs(rnorm(n))
  u<-runif(n,min(tim),max(tim))
  y<-pmin(tim,u)
   ic<-1*(tim<u)
m <- median(tim)
x<-matrix(rnorm(p*n),ncol=n)
  x[1:100, tim>m] <-  x[1:100, tim>m]+3
  return(list(x=x,y=y,ic=ic))
}
```

```
# generate training data; 2000 genes, 100 samples

junk<-gendata(n=100)
y<-junk$y
ic<-junk$ic
x<-junk$x
d <- list(x=x,survival.time=y, censoring.status=ic,
          geneid=as.character(1:nrow(x)),
          genenames=paste("g", as.character(1:nrow(x)), sep=""))

# train model
a3<- pamr.train(d, ngroup.survival=2)

#make class predictions

yhat <- pamr.predict(a3,d$x, threshold=1.0)

pamr.plotsurvival(yhat, d$survival.time, d$censoring.status)
```

`pamr.predictmany` *A function giving prediction information for many threshold values, from a nearest shrunken centroid fit.*

Description

A function giving prediction information for many threshold values, from a nearest shrunken centroid fit

Usage

```
pamr.predictmany(fit, newx, threshold=fit$threshold, prior =fit$prior, threshold
        threshold.scale, ...)
```

Arguments

`fit`	The result of a call to pamr.train
`newx`	Matrix of features at which predictions are to be made
`threshold`	The desired threshold values
`prior`	Prior probabilities for each class. Default is that specified in "fit"
`threshold.scale`	Additional scaling factors to be applied to the thresholds. Vector of length equal to the number of classes. Default is that specified in "fit".
`...`	Additional arguments to be passed to pamr.predict

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)

pamr.predictmany(mytrain, mydata$x)
```

`pamr.predict` *A function giving prediction information, from a nearest shrunken centroid fit.*

Description

A function giving prediction information, from a nearest shrunken centroid fit.

Usage

```
pamr.predict(fit, newx, threshold, type= c("class", "posterior", "centroid", "no
```

Arguments

`fit`	The result of a call to pamr.train
`newx`	Matrix of features at which predictions are to be made
`threshold`	The desired threshold value
`type`	Type of prediction desired: class predictions, posterior probabilities, (unshrunken) class centroids, vector of genes surviving the threshold
`prior`	Prior probabilities for each class. Default is that specified in "fit"
`threshold.scale`	Additional scaling factors to be applied to the thresholds. Vector of length equal to the number of classes. Default is that specified in "fit".

Details

`pamr.predict` Give a cross-tabulation of true versus predicted classes for the fit returned by pamr.train or pamr.cv, at the specified threshold

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=y)
mytrain <-   pamr.train(mydata)
mycv <- pamr.cv(mytrain,mydata)
pamr.predict(mytrain, mydata$x , threshold=1)
```

`pamr.surv.to.class2`

A function to assign observations to categories, based on their survival times.

Description

A function to assign observations to categories, based on their survival times.

Usage

```
pamr.surv.to.class2(y, icens, cutoffs=NULL, n.class=NULL,  class.names=NULL, new
```

Arguments

`y`	vector of survival times
`icens`	Vector of censorng status values: 1=died, 0=censored
`cutoffs`	Survival time cutoffs for categories. Default NULL
`n.class`	Number of classes to create: if cutoffs is NULL, n.class equal classes are created.
`class.names`	Character names for classes
`newy`	New set of survival times, for which probabilities are computed (see below). Default is y
`newic`	New set of censoring statuses, for which probabilities are computed (see below). Default is icens

Details

`pamr.pamr.surv.to.class2` splits observations into categories based on their survival times and the Kaplan-Meier estimates. For example if n.class=2, it makes two categories, one below the median survival, the other above. For each observation (newy, ic), it then computes the probability of that observation falling in each category. For an uncensored observation that probability is just 1 or 0 depending on when the death occurred. For a censored observation, the probabilities are based on the Kaplan Meier and are typically between 0 and 1.

Value

`class`	The category labels
`prob`	The estimates class probabilities
`cutoffs`	The cutoffs used

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
gendata<-function(n=100, p=2000){
  tim <- 3*abs(rnorm(n))
  u<-runif(n,min(tim),max(tim))
  y<-pmin(tim,u)
   ic<-1*(tim<u)
m <- median(tim)
x<-matrix(rnorm(p*n),ncol=n)
  x[1:100, tim>m] <-  x[1:100, tim>m]+3
  return(list(x=x,y=y,ic=ic))
}

# generate training data; 2000 genes, 100 samples

junk<-gendata(n=100)
y<-junk$y
ic<-junk$ic
x<-junk$x
d <- list(x=x, survival.time=y, censoring.status=ic,
          geneid=as.character(1:nrow(x)),
          genenames=paste("g", as.character(1:nrow(x)), sep=""))

# train model
a3<- pamr.train(d, ngroup.survival=2)

# generate test data
junkk<- gendata(n=500)

dd <- list(x=junkk$x, survival.time=junkk$y, censoring.status=junkk$ic)

# compute soft labels
proby <-  pamr.surv.to.class2(dd$survival.time, dd$censoring.status,
             n.class=a3$ngroup.survival)$prob
```

`pamr.test.errors.surv.compute`

A function giving a table of true versus predicted values, from a nearest shrunken centroid fit from survival data.

Description

A function giving a table of true versus predicted values, from a nearest shrunken centroid fit from survival data.

Usage

```
pamr.test.errors.surv.compute(proby, yhat)
```

Arguments

`proby`	Survival class probabilities, from pamr.surv.to.class2
`yhat`	Estimated class labels, from pamr.predict

Details

`pamr.test.errors.surv.compute` computes the erros between the true 'soft" class labels proby and the estimated ones "yhat"

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

Examples

```
gendata<-function(n=100, p=2000){
  tim <- 3*abs(rnorm(n))
  u<-runif(n,min(tim),max(tim))
  y<-pmin(tim,u)
   ic<-1*(tim<u)
m <- median(tim)
x<-matrix(rnorm(p*n),ncol=n)
  x[1:100, tim>m] <-  x[1:100, tim>m]+3
  return(list(x=x,y=y,ic=ic))
}

# generate training data; 2000 genes, 100 samples

junk<-gendata(n=100)
y<-junk$y
ic<-junk$ic
x<-junk$x
d <- list(x=x, survival.time=y, censoring.status=ic,
          geneid=as.character(1:nrow(x)),
          genenames=paste("g", as.character(1:nrow(x)), sep=""))

# train model
a3<- pamr.train(d, ngroup.survival=2)

# generate test data
junkk<- gendata(n=500)

dd <- list(x=junkk$x, survival.time=junkk$y, censoring.status=junkk$ic)

# compute soft labels
proby <-  pamr.surv.to.class2(dd$survival.time, dd$censoring.status,
             n.class=a3$ngroup.survival)$prob

# make class predictions for test data
yhat <- pamr.predict(a3,dd$x, threshold=1.0)

# compute test errors

pamr.test.errors.surv.compute(proby, yhat)
```

pamr.to.excel *A function to write out a data object into a tab-delimited text file*

Description

A function to write out a data object into a tab-delimited text file

Usage

```
pamr.to.excel(data,file,trace=TRUE)
```

Arguments

data	A data object, of the same form as is read in by pamr.from.excel. Must have components x (the matrix of expression values), y- a vector of class labels for each sample, geneid- a vector of gene identifiers and genenames- a vector of gene names. Optional components: samplelabels and batchlabels, both character vectors
file	Character name of a text file.
trace	Optional argument. If true, progress in writing out file is reported.

Details

pamr.to.excel writes out the data object into a tab-delimited text file, of the same form as is read in by pamr.from.excel. Useful for writing out data that has been imputed by pamr.knnimpute or adjusted by pamr.batchadjust. Note- this function writes the file out one line at a time, and hence can take a while for big datasets.

Author(s)

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

pamr.train *A function to train a nearest shrunken centroid classifier*

Description

A function that computes a nearest shrunken centroid for gene expression (microarray) data

Usage

```
pamr.train(data, gene.subset=NULL, sample.subset=NULL,
         threshold = NULL, n.threshold = 30,
        scale.sd = TRUE, threshold.scale = NULL, se.scale = NULL, offset.percent
         hetero=NULL, prior = NULL, remove.zeros = TRUE, sign.contrast="both",
ngroup.survival = 2)
```

Arguments

data	The input data. A list with components: x- an expression genes in the rows, samples in the columns), and y- a vector of the class labels for each sample. Optional components- genenames, a vector of gene names, and geneid- a vector of gene identifiers.
gene.subset	Subset of genes to be used. Can be either a logical vector of length total number of genes, or a list of integers of the row numbers of the genes to be used
sample.subset	Subset of samples to be used. Can be either a logical vector of length total number of samples, or a list of integers of the column numbers of the samples to be used.
threshold	A vector of threshold values for the centroid shrinkage. Default is a set of 30 values chosen by the software
n.threshold	Number of threshold values desired (default 30)
scale.sd	Scale each threshold by the wthin class standard deviations? Default: true
threshold.scale	Additional scaling factors to be applied to the thresholds. Vector of length equal to the number of classes. Default- a vectors of ones.
se.scale	Vector of scaling factors for the within class standard errors. Default is sqrt(1/n.class-1/n), where n is the overall sample size and n.class is the sample sizes in each class. This default adjusts for different class sizes.
offset.percent	Fudge factor added to the denominator of each t-statistic, expressed as a percentile of the gene standard deviation values. This is a small positive quantity to penalize genes with expression values near zero, which can result in very large ratios. This factor is expecially impotant for Affy data. Default is the median of the standard deviations of each gene.
hetero	Should a heterogeneity transformation be done? If yes, hetero must be set to one of the class labels (see Details below). Default is no (hetero=NULL)
prior	Vector of length the number of classes, representing prior probabilities for each of the classes. The prior is used in Bayes rule for making class prediction. Default is NULL, and prior is then taken to be n.class/n, where n is the overall sample size and n.class is the sample sizes in each class.
remove.zeros	Remove threshold values yielding zero genes? Default TRUE
sign.contrast	Directions of allowed deviations of class-wise average gene expression from the overall average gene expression. Default is “both” (positive or negative). Can also be set to “positive” or “negative”.
ngroup.survival	Number of groups formed for survival data. Default 2

Details

pamr.train fits a nearest shrunken centroid classifier to gene expression data. Details may be found in the PNAS paper referenced below. One feature not described there is "heterogeneity analysis". Suppose there are two classes labelled "A" and "B". CLass "A" is considered a normal class, and "B" an abnormal class. Setting hetero="A" transforms expression values x[i,j] to |x[i,j]-mean(x[i,j])| where the mean is taken only over samples in class "A". The transformed feature values are then used in Pam. This is useful when the abnormal class "B" is heterogeneous, i.e. a

given gene might have higher expresion than normal for some class "B" samples, and lower for others. With more than 2 classes, each class is centered on the class specified by hetero.

Value

A list with components

`y`	The outcome classes.
`yhat`	A matrix of predicted classes, each column representing the results from one threshold.
`prob`	A array of predicted class probabilities. of dimension n by nclass by n.threshold. n is the number samples, nclass is the number of classes, n.threshold is the number of thresholds tried
`centroids`	A matrix of (unshrunken) class centroids, n by nclass
`hetero`	Value of hetero used in call to pamr.train
`norm.cent`	Centroid of "normal" group, if hetero was specified
`centroid.overall`	A vector containing the (unshrunken) overall centroid (all classes together)
`sd`	A vector of the standard deviations for each gene
`threshold`	A vector of the threshold tried in the shrinkage
`nonzero`	A vector of the number of genes that survived the thresholding, for each threshold value tried
`threshold.scale`	A vector of threshold scale factors that were used
`se.scale`	A vector of standard error scale factors that were used
`call`	The calling sequence used
`prior`	The prior probabilities used
`errors`	The number of trainin errors for each threshold value

Author(s)

Trevor Hastie,Robert Tibshirani, Balasubramanian Narasimhan, and Gilbert Chu

References

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu Diagnosis of multiple cancer types by shrunken centroids of gene expression PNAS 99: 6567-6572. Available at www.pnas.org

Examples

```
#generate some data
set.seed(120)
x <- matrix(rnorm(1000*20),ncol=20)
y <- sample(c(1:4),size=20,replace=TRUE)
mydata <- list(x=x,y=factor(y))

#train classifier
results<-   pamr.train(mydata)

# train classifier on all  data except class 4
```

```
results2 <- pamr.train(mydata,sample.subset=(mydata$y!=4))

# train classifier on  only the first 500 genes
results3 <- pamr.train(mydata,gene.subset=1:500)
```

Index