runSAM  Run the sam webapp

**Description**

Runs the sam web application for a graphical user interface.

**Usage**

runSAM()

**Details**

Uses shiny to create a graphical user interface for SAM

**Author(s)**

Michael J. Seo

**References**


**Examples**

```r
## Not run: runSAM()
```

---

SAM  Significance analysis of microarrays - simple user interface

**Description**

Correlates a large number of features (eg genes) with an outcome variable, such as a group indicator, quantitative variable or survival time. This is a simple user interface for the samr function applied to array data. For sequencing data applications, see the function SAMseq.

**Usage**

```r
SAM(x,y=NULL,censoring.status=NULL, resp.type=c("Quantitative","Two class unpaired","Survival","Multiclass", "One class", "Two class paired","Two class unpaired timecourse", "One class timecourse","Two class paired timecourse", "Pattern discovery"), geneid = NULL, genenames = NULL, s0=NULL, 
```
s0.perc=NULL,  
nperms=100,  
center.arrays=FALSE,  
testStatistic=c("standard","wilcoxon"),  
time.summary.type=c("slope","signed.area"),  
regression.method=c("standard","ranks"),  
return.x=TRUE,  
knn.neighbors=10,  
random.seed=NULL,  
logged2 = FALSE,  
fdr.output = 0.20,  
eigengene.number = 1)

Arguments

x Feature matrix: p (number of features) by n (number of samples), one observation per column (missing values allowed)

y n-vector of outcome measurements

censoring.status n-vector of censoring censoring.status (1= died or event occurred, 0=survived, or event was censored), needed for a censored survival outcome

resp.type Problem type: "Quantitative" for a continuous parameter; "Two class unpaired"; "Survival" for censored survival outcome; "Multiclass": more than 2 groups; "One class" for a single group; "Two class paired" for two classes with paired observations; "Two class unpaired timecourse", "One class time course", "Two class.paired timecourse", "Pattern discovery"

geneid Optional character vector of geneids for output.

genenames Optional character vector of genenames for output.

s0 Exchangeability factor for denominator of test statistic; Default is automatic choice. Only used for array data.

s0.perc Percentile of standard deviation values to use for s0; default is automatic choice; -1 means s0=0 (different from s0.perc=0, meaning s0=zeroeth percentile of standard deviation values= min of sd values. Only used for array data.

nperms Number of permutations used to estimate false discovery rates

center.arrays Should the data for each sample (array) be median centered at the outset? Default =FALSE. Only used for array data.

testStatistic Test statistic to use in two class unpaired case. Either "standard" (t-statistic) or "wilcoxon" (Two-sample wilcoxon or Mann-Whitney test). Only used for array data.

time.summary.type Summary measure for each time course: "slope", or "signed.area"). Only used for array data.

regression.method Regression method for quantitative case: "standard", (linear least squares) or "ranks" (linear least squares on ranked data). Only used for array data.
return.x    Should the matrix of feature values be returned? Only useful for time course 
data, where x contains summaries of the features over time. Otherwise x is the 
same as the input data data$x

knn.neighbors    Number of nearest neighbors to use for imputation of missing features values. 
Only used for array data.

random.seed    Optional initial seed for random number generator (integer)

logged2    Has the data been transformed by log (base 2)? This information is used only 
for computing fold changes

fdr.output    (Approximate) False Discovery Rate cutoff for output in significant genes table

eigengene.number    Eigengene to be used (just for resp.type="Pattern discovery")

Details

This is a simple, user-friendly interface to the samr package used on array data. It calls samr, 
samr.compute.delta.table and samr.compute.siggenes.table. samr detects differential expression for 
microarray data, and sequencing data, and other data with a large number of features. samr is the R 
package that is called by the "official" SAM Excel Addin. The format of the response vector y and 
the calling sequence is illustrated in the examples below. A more complete description is given in 
the SAM manual at http://www-stat.stanford.edu/~tibs/SAM

Value

A list with components

samr.obj    Output of samr. See documentation for samr for details.

siggenes.table    Table of significant genes, output of samr.compute.siggenes.table. This has 
components: genes.up—matrix of significant genes having positive correla-
tion with the outcome and genes.lo—matrix of significant genes having nega-
tive correlation with the outcome. For survival data, genes.up are those genes 
having positive correlation with risk- that is, increased expression corresponds 
to higher risk (shorter survival) genes.lo are those whose increased expression 
corresponds to lower risk (longer survival).

delta.table    Output of samr.compute.delta.table.

del    Value of delta (distance from 45 degree line in SAM plot) for used for creating 
delta.table and siggenes.table. Changing the input value fdr.output will change 
the resulting del.

call    The calling sequence

Author(s)

Jun Li and Balasubramanian Narasimhan and Robert Tibshirani
References


Examples

########## two class unpaired comparison
# y must take values 1,2
set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)
u<-matrix(2+rnorm(100),ncol=10,nrow=100)
y<-c(rep(1,10),rep(2,10))
samfit<-SAM(x,y,resp.type="Two class unpaired")

# examine significant gene list
print(samfit)

# plot results
plot(samfit)

########### two class paired
# y must take values -1, 1, -2, 2 etc, with (-k,k) being a pair
set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)
u<-matrix(2+rnorm(100),ncol=10,nrow=100)
y<-c(rep(-1,10),c(1:10))
samfit<-SAM(x,y, resp.type="Two class paired",fdr.output=.25)

############################quantitative response
set.seed(30)
p=1000
x<-matrix(rnorm(p*20),ncol=20)
y<-rnorm(20)
x[1:20,y>0]=x[1:20,y>0]+4
a<-SAM(x,y,resp.type="Quantitative",nperms=50,fdr.output=.5)

survival data
# y is numeric; censoring.status=1 for failures, and 0 for censored
set.seed(84048)
x=matrix(rnorm(1000*50),ncol=50)
x[1:50,26:50]=x[1:50,26:50]+2
x[51:100,26:50]=x[51:100,26:50]-2
y=abs(rnorm(50))
y[26:50]=y[26:50]+2
censoring.status <- sample(c(0,1),size=50,replace=TRUE)
a<-SAM(x,y,censoring.status=censoring.status,resp.type="Survival",nperms=20)

multi-class example
# y takes values 1,2,3,...k where k= number of classes
set.seed(84048)
x=matrix(rnorm(1000*10),ncol=10)
y=c(rep(1,3),rep(2,3),rep(3,4))
a <- SAM(x,y,resp.type="Multiclass",nperms=50)

pattern discovery
# here there is no outcome y; the desired eigengene is indicated by
# the argument eigengene.numbe in the data object
set.seed(32)
x=matrix(rnorm(1000*9),ncol=9)
mu=c(3,2,1,0,0,0,1,2,3)
samr

Significance analysis of microarrays

Description

Correlates a large number of features (eg genes) with an outcome variable, such as a group indicator, quantitative variable or survival time. NOTE: for most users, the interface function SAM— which

```r
b=3*runif(100)+.5
x[1:100,]=x[1:100,]+ b
d=list(x=x,eigengene.number=1,
geneid=as.character(1:nrow(x)),genenames=paste("gene",as.character(1:nrow(x))))
a <- SAM(x, resp.type="Pattern discovery", nperms=50)

############################ timecourse data
# elements of y are of the form ktimet where k is the class label and t
# is the time; in addition, the suffixes Start or End indicate the first
# and last observation in a given time course
# the class label can be that for a two class unpaired, one class or
# two class paired problem
set.seed(8332)
y=paste(c(rep(QLQU),rep(2L15)),"Time",rep(c(1L2L3L4L5L1L2L3L5L6L7L8),3),
sep="")
start=c(1L6L11L16L21L26)
for(i in start){
y[i]=paste(y[i],"Start",sep="")
}
for(i in start+4){
y[i]=paste(y[i],"End",sep="")
}
x=matrix(rnorm(PPP*SP),ncol=SP)
x{Q:UPLQ6:RP}x{Q:UPLQ6:RP}+matrix(S*c(PLQLRLSLT),ncol=ULnrow=UPLbyrow=true)
x{UQ:QPPLQ6:RP}Mmatrix(S*c(PLQLRLSLT),ncol=ULnrow=UPLbyrow=true)
x{UQ:QPPLRQ:RU}Mmatrix(S*c(PLQLRLSLT),ncol=ULnrow=UPLbyrow=true)
x{UQ:QPPLR6:SP}Mmatrix(S*c(PLQLRLSLT),ncol=ULnrow=UPLbyrow=true)

```

```r
x[1:50,16:20]=x[1:50,16:20]+matrix(3*c(0L1L2L3L4),ncol=5,nrow=50,byrow=TRUE)
x[1:50,26:30]=x[1:50,26:30]+matrix(3*c(0L1L2L3L4),ncol=5,nrow=50,byrow=TRUE)
x[51:100,16:20]=x[51:100,16:20]-matrix(3*c(0L1L2L3L4),ncol=5,nrow=50,byrow=TRUE)
x[51:100,21:25]=x[51:100,21:25]-matrix(3*c(0L1L2L3L4),ncol=5,nrow=50,byrow=TRUE)
x[51:100,26:30]=x[51:100,26:30]-matrix(3*c(0L1L2L3L4),ncol=5,nrow=50,byrow=TRUE)
```

```r
a<- SAM(x,y, resp.type="Two class unpaired timecourse", nperms=100, time.summary.type="slope")
```
calls samr– will be more convenient for array data, and the interface function SAMseq– which also calls samr– will be more convenient for sequencing data.

Usage

samr(data, resp.type=c("Quantitative","Two class unpaired","Survival","Multiclass","One class","Two class paired","Two class unpaired timecourse","One class timecourse","Two class paired timecourse","Pattern discovery"), assay.type=c("array","seq"), s0=NULL, s0.perc=NULL, nperms=100, center.arrays=FALSE, testStatistic=c("standard","wilcoxon"), time.summary.type=c("slope","signed.area"), regression.method=c("standard","ranks"), return.x=FALSE, knn.neighbors=10, random.seed=NULL, nresamp=20, nresamp.perm=NULL, xl.mode=c("regular","firsttime","next20","lasttime"), xl.time=NULL, xl.prevfit=NULL)

Arguments

data

Data object with components x- p by n matrix of features, one observation per column (missing values allowed); y- n-vector of outcome measurements; censoring.status- n-vector of censoring, 1=died or event occurred, 0=survived, or event was censored), needed for a censored survival outcome

resp.type

Problem type: "Quantitative" for a continuous parameter (Available for both array and sequencing data); "Two class unpaired" (for both array and sequencing data); "Survival" for censored survival outcome (for both array and sequencing data); "Multiclass": more than 2 groups (for both array and sequencing data); "One class" for a single group (only for array data); "Two class paired" for two classes with paired observations (for both array and sequencing data); "Two class unpaired timecourse" (only for array data), "One class time course" (only for array data), "Two class.paired timecourse" (only for array data), or "Pattern discovery" (only for array data)

assay.type

Assay type: "array" for microarray data, "seq" for counts from sequencing

s0

Exchangeability factor for denominator of test statistic; Default is automatic choice. Only used for array data.

s0.perc

Percentile of standard deviation values to use for s0; default is automatic choice; -1 means s0=0 (different from s0.perc=0, meaning s0=zeroeth percentile of standard deviation values= min of sd values. Only used for array data.

nperms

Number of permutations used to estimate false discovery rates

center.arrays

Should the data for each sample (array) be median centered at the outset? Default =FALSE. Only used for array data.

testStatistic

Test statistic to use in two class unpaired case.Either "standard" (t-statistic) or ,"wilcoxon" (Two-sample wilcoxon or Mann-Whitney test). Only used for array data.

time.summary.type

Summary measure for each time course: "slope", or "signed.area"). Only used for array data.
**regression.method**
Regression method for quantitative case: "standard", (linear least squares) or "ranks" (linear least squares on ranked data). Only used for array data.

**return.x**
Should the matrix of feature values be returned? Only useful for time course data, where x contains summaries of the features over time. Otherwise x is the same as the input data data$x.

**knn.neighbors**
Number of nearest neighbors to use for imputation of missing features values. Only used for array data.

**random.seed**
Optional initial seed for random number generator (integer)

**nresamp**
For assay.type="seq", number of resamples used to construct test statistic. Default 20. Only used for sequencing data.

**nresamp.perm**
For assay.type="seq", number of resamples used to construct test statistic for permutations. Default is equal to nresamp and it must be at most nresamp. Only used for sequencing data.

**xl.mode**
Used by Excel interface

**xl.time**
Used by Excel interface

**xl.prevfit**
Used by Excel interface

**Details**
Carries out a SAM analysis. Applicable to microarray data, sequencing data, and other data with a large number of features. This is the R package that is called by the "official" SAM Excel package v2.0. The format of the response vector y and the calling sequence is illustrated in the examples below. A more complete description is given in the SAM manual at http://www-stat.stanford.edu/~tibs/SAM

**Value**
A list with components

**n**
Number of observations

**x**
Data matrix p by n (p=# genes or features). Equal to the matrix data$x in the original call to samr except for (1) time course analysis, where is contains the summarized data or (2) quantitative outcome with rank regression, where it contains the data transformed to ranks. Hence it is null except for in time course analysis.

**y**
Vector of n outcome values. equal the values data$y in the original call to samr, except for (1) time course analysis, where is contains the summarized y or (2) quantitative outcome with rank regression, where it contains the y values transformed to ranks.

**argy**
The values data$y in the original call to samr

**censoring.status**
Censoring status indicators if applicable

**testStatistic**
Test Statistic used
nnperms Number of permutations requested
nperms.act Number of permutations actually used. Will be < nperms when \# of possible permutations <= nperms (in which case all permutations are done)
ttt=nnumer/sd, the vector of p test statistics for original data
tnumer Numerators for tt
tsddenominators for tt. Equal to standard deviation for feature plus s0
s0 Computed exchangeability factor
s0.perc Computed percentile of standard deviation values. s0= s0.perc percentile of the gene standard deviations
eva P-vector of expected values for tt under permutation sampling
perms nperms.act by n matrix of permutations used. Each row is a permutation of 1,2...nnpermssy nperms.act by n matrix of permutations used. Each row is a permutation of y1,y2,...yn. Only one of perms or permssy is non-Null, depending on resp.type
all.perms.flag Were all possible permutations used?
ttt* p by nperms.aca matrix t of test statistics from permuted data. Each column if sorted in ascending order	ttstar0 p by nperms.act matrix of test statistics from permuted data. Columns are in order of data
eigengene.number The number of the eigengene (eg 1,2,...) that was requested for Pattern discovery
eigengene Computed eigengene
pi0 Estimated proportion of non-null features (genes)
foldchange P-vector of foldchanges for original data
foldchange.star p by nperms.act matrix estimated foldchanges from permuted data
sdstar.keep n by nperms.act matrix of standard deviations from each permutation
censoring.status.star.keep n by nperms.act matrix of censoring.status indicators from each permutation
resp.type The response type used. Same as resp.type.arg, except for time course data, where time data is summarized and then treated as non-time course. Eg if resp.type.arg="oneclass.timecourse" then resp.type="oneclass"
resp.type.arg The response type requested in the call to samr
stand.contrasts For multiclass data, p by nclass matrix of standardized differences between the class mean and the overall mean
stand.contrasts.star For multiclass data, p by nclass by nperms.act array of standardized contrasts for permuted datasets
stand.contrasts.95 For multiclass data, 2.5 of standardized contrasts. Useful for determining which class contrast for significant genes, are large
depth For array.type="seq", estimated sequencing depth for each sample.
call calling sequence
Author(s)
Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

References
Li, Jun and Tibshirani, R. (2011). Finding consistent patterns: a nonparametric approach for identi-
fying differential expression in RNA-Seq data. To appear, Statistical Methods in Medical Research.

Examples

####### two class unpaired comparison
# y must take values 1,2

set.seed(100)
x <- matrix(rnorm(1000*20),ncol=20)
dd <- sample(1:1000, size=100)

u <- matrix(2*rnorm(100), ncol=10, nrow=100)
x[dd,1:20] <- x[dd,1:20] + u
y <- c(rep(1,10), rep(2,10))

data <- list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g",as.character(1:nrow(x)), sep=""), logged2=TRUE)

samr.obj <- samr(data, resp.type="Two class unpaired", nperms=100)
delta = .4
samr.plot(samr.obj, delta)
delta.table <- samr.compute.delta.table(samr.obj)
siggenes.table <- samr.compute.siggenes.table(samr.obj, delta, data, delta.table)

# sequence data

set.seed(3)
x <- abs(100 * matrix(rnorm(1000*20), ncol=20))
x <- trunc(x)
y <- c(rep(1,10), rep(2,10))
x[1:50, y==2] <- x[1:50, y==2] + 50
data <- list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g",as.character(1:nrow(x)), sep=""))

samr.obj <- samr(data, resp.type="Two class unpaired", assay.type="seq", nperms=100)
delta = 5
samr.plot(samr.obj, delta)
delta.table <- samr.compute.delta.table(samr.obj)
siggenes.table <- samr.compute.siggenes.table(samr.obj, delta, data, delta.table)

################ two class paired

# y must take values -1, 1, -2, 2 etc, with (-k,k) being a pair

set.seed(100)
x <- matrix(rnorm(1000*20), ncol=20)
dd <- sample(1:1000, size=100)
u <- matrix(2*rnorm(100), ncol=10, nrow=100)
x[dd, 1:20] <- x[dd, 1:20] + u
y <- c(-1:10, 1:10)

d <- list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g", as.character(1:nrow(x)), sep=""), loggedR=TRUE)
samr.obj <- samr(d, resp.type="Two class paired", nperms=100)

################### quantitative response

# y must take numeric values

set.seed(84048)
x <- matrix(rnorm(1000*9), ncol=9)
mu <- c(3, 2, 1, 0, 0, 1, 2, 3)
b <- runif(100) + .5
x[1:100, ] <- x[1:100, ] + b
y <- mu

d <- list(x=x, y=y,
genenamest=as.character(1:nrow(x)), genenames=paste("gene", as.character(1:nrow(x))))
samr.obj = samr(d, resp.type="Quantitative", nperms=50)

##################### oneclass

# y is a vector of ones
```r
set.seed(100)
x <- matrix(rnorm(1000*20), ncol=20)
dd <- sample(1:1000, size=100)

c <- matrix(2*rnorm(100), ncol=10, nrow=100)
x[dd, 11:20] <- x[dd, 11:20] + c

y <- c(rep(1, 20))
data <- list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g", as.character(1:nrow(x)), sep=""),
logged=TRUE)
samr.obj <- samr(data, resp.type="One class", nperms=100)

### survival data
# y is numeric; censoring.status=1 for failures, and 0 for censored
set.seed(84048)
x <- matrix(rnorm(1000*50), ncol=50)
x[1:50, 26:50] <- x[1:50, 26:50] + 2
x[51:100, 26:50] <- x[51:100, 26:50] - 2
y <- abs(rnorm(50))
y[26:50] <- y[26:50] + 2
censoring.status <- sample(c(0, 1), size=50, replace=TRUE)
d <- list(x=x, y=y, censoring.status=censoring.status,
geneid=as.character(1:1000), genenames=paste("gene", as.character(1:1000)))
samr.obj <- samr(d, resp.type="Survival", nperms=20)

### multi-class example
# y takes values 1, 2, 3,...k where k= number of classes
set.seed(84048)
x <- matrix(rnorm(1000*10), ncol=10)
x[1:50, 6:10] <- x[1:50, 6:10] + 2
x[51:100, 6:10] <- x[51:100, 6:10] - 2

c <- c(rep(1, 3), rep(2, 3), rep(3, 4))
d <- list(x=x, y=y, geneid=as.character(1:1000),
genenames=paste("gene", as.character(1:1000)))
samr.obj <- samr(d, resp.type="Multiclass")

### timecourse data
# elements of y are of the form kTimet where k is the class label and t
# is the time; in addition, the suffixes Start or End indicate the first and last observation in a given time course
# the class label can be that for a two class unpaired, one class or two class paired problem

set.seed(8332)
y=paste(c(rep(1,15),rep(2,15)),"Time",rep(c(1,2,3,4,5,1,2,5,3.7,4.1,5.5),3), sep="")
start=c(1,6,11,16,21,26)
for(i in start){
y[i]=paste(y[i],"Start",sep="")
}
for(i in start+4){
y[i]=paste(y[i],"End",sep="")
}
x=matrix(rnorm(9*30),ncol=30)
x[1:50,16:20]=x[1:50,16:20]+matrix(3*c(0,1,2,3,4),ncol=5,nrow=50,byrow=TRUE)
x[1:50,21:25]=x[1:50,21:25]+matrix(3*c(0,1,2,3,4),ncol=5,nrow=50,byrow=TRUE)
x[1:50,26:30]=x[1:50,26:30]+matrix(3*c(0,1,2,3,4),ncol=5,nrow=50,byrow=TRUE)

x[51:100,16:20]=x[51:100,16:20]-matrix(3*c(0,1,2,3,4),ncol=5,nrow=50,byrow=TRUE)
x[51:100,21:25]=x[51:100,21:25]-matrix(3*c(0,1,2,3,4),ncol=5,nrow=50,byrow=TRUE)
x[51:100,26:30]=x[51:100,26:30]-matrix(3*c(0,1,2,3,4),ncol=5,nrow=50,byrow=TRUE)

data=list(x=x,y=y, geneid=as.character(1:nrow(x)), genenames=paste("g",as.character(1:nrow(x)),sep=""), logged2=TRUE)

samr.obj<- samr(data, resp.type="Two class unpaired timecourse", nperms=100, time.summary.type="slope")

#################################################### pattern discovery
# here there is no outcome y; the desired eigengene is indicated by the argument eigengene.numero in the data object

set.seed(32)
x=matrix(rnorm(9*9),ncol=9)
mu=c(3,2,1,0,0,1,2,3)
b=3*runif(100)+.5
x[1:100,]=x[1:100,]+ b

d=list(x=x, eigengene.number=1, geneid=as.character(1:nrow(x)), genenames=paste("gene", as.character(1:nrow(x))))

samr.obj=samr(d, resp.type="Pattern discovery", nperms=50)
**samr.assess.samplesize**

*Assess the sample size for a SAM analysis*

**Description**

Estimate the false discovery rate, false negative rate, power and type I error for a SAM analysis. Currently implemented only for two class (unpaired or paired), one-sample and survival problems.

**Usage**

```r
samr.assess.samplesize(samr.obj, data, dif, samplesize.factors=c(1,2,3,5),
min.genes = 10, max.genes = nrow(data$x)/2)
```

**Arguments**

- **samr.obj**: Object returned from call to samr
- **data**: Data list, same as that passed to samr.train
- **dif**: Change in gene expression between groups 1 and 2, for genes that are differentially expressed. For log base 2 data, a value of 1 means a 2-fold change. For One-sample problems, dif is the number of units away from zero for differentially expressed genes. For survival data, dif is the numerator of the Cox score statistic (this info is provided in the output of samr).
- **samplesize.factors**: Integer vector of length 4, indicating the sample sizes to be examined. The values are factors that multiply the original sample size. So the value 1 means a sample size of ncol(data$x), 2 means a sample size of ncol(data$x), etc.
- **min.genes**: Minimum number of genes that are assumed to truly changed in the population
- **max.genes**: Maximum number of genes that are assumed to truly changed in the population

**Details**

Estimates false discovery rate, false negative rate, power and type I error for a SAM analysis. The argument samplesize.factor allows the use to assess the effect of varying the sample size (total number of samples). A detailed description of this calculation is given in the SAM manual at [http://www-stat.stanford.edu/~tibs/SAM](http://www-stat.stanford.edu/~tibs/SAM)

**Value**

A list with components

- **Results**: A matrix with columns: number of genes- both the number differentially expressed genes in the population and number called significant; cutpoint- the threshold used for the absolute SAM score d; FDR, 1-power- the median false discovery rate, also equal to the power for each gene; FDR-90perc- the upper 90th percentile of the FDR; FNR, Type 1 error- the false negative rate, also equal
to the type I error for each gene; FNR-90perc- the upper 90th percentile of the FNR

dif.call Change in gene expression between groups 1 and 2, that was provided in the call to samr.assess.samplesize
difm The average difference in SAM score d for the genes differentially expressed vs unexpressed
samplesize.factor The samplesize.factor that was passed to samr.assess.samplesize
n Number of samples in input data (i.e. ncol of x component in data)

Author(s)
Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

References
A more complete description is given in the SAM manual at http://www-stat.stanford.edu/~tibs/SAM

Examples
#generate some example data
set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)
u<-matrix(2*runif(100),ncol=10,nrow=100)
x[dd,1:20]<-x[dd,1:20]+u
y<-c(rep(1,10),rep(2,10))
data=list(x=x,y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g",as.character(1:nrow(x)),sep=""), logged=TRUE)
log2=function(x){log(x)/log(2)}

# run SAM first
samr.obj<-samr(data, resp.type="Two class unpaired", nperms=100)

# assess current sample size (20), assuming 1.5fold difference on log base 2 scale
samr.assess.samplesize.obj<-samr.assess.samplesize(samr.obj, data, log2(1.5))

# assess the effect of doubling the sample size
samr.assess.samplesize.obj2<-samr.assess.samplesize(samr.obj, data, log2(1.5))
Make a plot of the results from `samr.assess.samplesize`

**Description**

Plots of the results from `samr.assess.samplesize`

**Usage**

```r
samr.assess.samplesize.plot(samr.assess.samplesize.obj, logx=TRUE)
```

**Arguments**

- `samr.assess.samplesize.obj`: Object returned from call to `samr.assess.samplesize`
- `logx`: Should logs be used on the horizontal (# of genes) axis? Default TRUE

**Details**

Plots results: FDR (or 1-power) and FNR (or 1-type 1 error) from `samr.assess.samplesize`

**Author(s)**

Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

**References**


**Examples**

```r
# generate some example data
set.seed(100)
x <- matrix(rnorm(1000*20), ncol=20)
dd <- sample(1:1000, size=100)

u <- matrix(2*runif(100), ncol=10, nrow=100)

y <- c(rep(1, 10), rep(2, 10))
data = list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g", as.character(1:nrow(x)), sep=""), logged=TRUE)

log2 = function(x){log(x)/log(2)}

# run SAM first
```
```
samr.obj<-samr(data, resp.type="Two class unpaired", nperms=100)

# assess current sample size (20), assuming 1.5 fold difference on the log base 2 scale
samr.assess.samplesize.obj<- samr.assess.samplesize(samr.obj, data, log2(1.5))
samr.assess.samplesize.plot(samr.assess.samplesize.obj)
```

---

**samr.compute.delta.table**

*Compute delta table for SAM analysis*

**Description**

Computes tables of thresholds, cutpoints and corresponding False Discovery rates for SAM (Significance analysis of microarrays) analysis

**Usage**

```r
samr.compute.delta.table(samr.obj, min.foldchange=0, dels=NULL, nvals=50)
```

**Arguments**

- `samr.obj`: Object returned from call to samr
- `min.foldchange`: The minimum fold change desired; should be >1; default is zero, meaning no fold change criterion is applied
- `dels`: vector of delta values used. Delta is the vertical distance from the 45 degree line to the upper and lower parallel lines that define the SAM threshold rule. By default, for array data, 50 values are chosen in the relevant operating change for delta. For sequencing data, the maximum number of effective delta values are chosen automatically according to the data.
- `nvals`: Number of delta values used. For array data, the default value is 50. For sequencing data, the value will be chosen automatically.

**Details**

Returns a table of the FDR and upper and lower cutpoints for various values of delta, for a SAM analysis.

**Author(s)**

Balasubramanian Narasimhan and Robert Tibshirani

**References**

Examples

```r
#generate some example data
set.seed(100)
x <- matrix(rnorm(100*20), ncol=20)
dd <- sample(1:1000, size=100)

u <- matrix(2*runif(100), ncol=10, nrow=100)
x[dd, 1:20] <- x[dd, 1:20] + u

y <- c(rep(1, 10), rep(2, 10))
data = list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g", as.character(1:nrow(x)), sep=""), logged=TRUE)

samr.obj <- samr(data, resp.type="Two class unpaired", nperms=50)
delta.table <- samr.compute.delta.table(samr.obj)
```

---

**samr.compute.siggenes.table**

*Compute significant genes table*

Description

Computes significant genes table, starting with samr object "samr.obj" and delta.table "delta.table"

Usage

```r
samr.compute.siggenes.table(samr.obj, del, data, delta.table, 
min.foldchange=0, all.genes=FALSE, compute.localfdr=FALSE)
```

Arguments

- **samr.obj**: Object returned from call to samr
- **del**: Value of delta to define cutoff rule
- **data**: Data object, same as that used in call to samr
- **delta.table**: Object returned from call to samr.compute.delta.table
- **min.foldchange**: The minimum fold change desired; should be >1; default is zero, meaning no fold change criterion is applied
- **all.genes**: Should all genes be listed? Default FALSE
- **compute.localfdr**: Should the local fdrs be computed (this can take some time)? Default FALSE
Value

return(list(genes.up=res.up, genes.lo=res.lo, color.ind.for.multi=color.ind.for.multi, ngenes.up=ngenes.up, ngenes.lo=ngenes.lo))

genes.up Matrix of significant genes having positive correlation with the outcome. For survival data, genes.up are those genes having positive correlation with risk- that is, increased expression corresponds to higher risk (shorter survival).

genes.lo Matrix of significant genes having negative correlation with the outcome. For survival data, genes.lo are those whose increased expression corresponds to lower risk (longer survival).

color.ind.for.multi For multiclass response: a matrix with entries +1 if the class mean is larger than the overall mean at the 95 levels, -1 if less, and zero otherwise. This is useful in determining which class or classes causes a feature to be significant

ngenes.up Number of significant genes with positive correlation

ngenes.lo Number of significant genes with negative correlation

Author(s)

Balasubramanian Narasimhan and Robert Tibshirani

References


Examples

#generate some example data
set.seed(100)
x<-matrix(rnorm(1000*20),ncol=20)
dd<-sample(1:1000,size=100)

u<-matrix(2*runorm(100),ncol=10,nrow=100)
y<-c(rep(1,10),rep(2,10))
data=list(x=x,y=y, geneid=as.character(1:nrow(x)), genenames=paste("g",as.character(1:nrow(x)),sep=""), loggedR=TRUE)

samr.obj<-samr(data, resp.type="Two class unpaired", nperms=100)
delta.table<-samr.compute.delta.table(samr.obj)
del<- 0.3
siggenes.table<- samr.compute.siggenes.table(samr.obj, del, data, delta.table)
**Description**

Estimate the sequencing depth of each experiment for sequencing data.

**Usage**

samr.estimate.depth(x)

**Arguments**

- **x**
  
  the original count matrix. p by n matrix of features, one observation per column.

**Details**

normalize the data matrix so that each number looks roughly like Gaussian distributed and each experiment has the same sequencing depth. To do this, we first use Anscombe transformation to stabilize the variance and makes each number look like Gaussian, and then divide each experiment by the square root of the sequencing depth.

**Value**

- **depth**
  
  sequencing depth of each experiment. a vector of length n.

**Author(s)**

Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

**References**


**Examples**

```r
set.seed(100)
mu <- matrix(100, 1000, 20)
mu[1:100, 11:20] <- 200
mu <- scale(mu, center=FALSE, scale=runif(20, 0.5, 1.5))
x <- matrix(rpois(length(mu), mu), 1000, 20)
y <- c(rep(1, 10), rep(2, 10))
data=list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g",as.character(1:nrow(x)),sep=""))
depth <- samr.estimate.depth(data$x)
```
Estimate the miss rate table for a SAM analysis

Description
Estimates the miss rate table, showing the local false negative rate, for a SAM analysis

Usage
samr.missrate(samr.obj, del, delta.table, quant=NULL)

Arguments
- **samr.obj**: Object returned from call to samr
- **del**: Value of delta to define cutoff rule
- **delta.table**: Object returned from call to samr.compute.delta.table
- **quant**: Optional vector of quantiles to used in the miss rate calculation

Author(s)
Jun Li and Balasubramian Narasimhan and Robert Tibshirani

References

Examples
#generate some example data
set.seed(100)
x <- matrix(rnorm(1000*20), ncol=20)
dd <- sample(1:1000, size=100)

u <- matrix(rnorm(100), ncol=10, nrow=10)

y <- c(rep(1,10), rep(2,10))
data=list(x=x, y=y, geneid=as.character(1:nrow(x)), genenames=paste("g", as.character(1:nrow(x)), sep=""), logged=TRUE)

samr.obj <- samr(data, resp.type="Two class unpaired", nperms=100)
delta.table <- samr.compute.delta.table(samr.obj)
**Description**

Output a normalized sequencing data matrix from the original count matrix.

**Usage**

```r
samr.norm.data(x, depth=NULL)
```

**Arguments**

- `x` the original count matrix. \( p \) by \( n \) matrix of features, one observation per column.
- `depth` sequencing depth of each experiment. a vector of length \( n \). This function will estimate the sequencing depth if it is not specified.

**Details**

Normalize the data matrix so that each number looks roughly like Gaussian distributed and each experiment has the same sequencing depth. To do this, we first use Anscombe transformation to stabilize the variance and makes each number look like Gaussian, and then divide each experiment by the square root of the sequencing depth.

**Value**

- `x` the normalized data matrix.

**Author(s)**

Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

**References**

Examples

```r
set.seed(100)
mu <- matrix(c(100, 1000, 20)
mu[1:100, 11:20] <- 200
mu <- scale(mu, center=FALSE, scale=runif(20, 0.5, 1.5))
x <- matrix(rpois(length(mu), mu), 1000, 20)
y <- c(rep(1, 10), rep(2, 10))
data=list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g", as.character(1:nrow(x)), sep=" "))
x.norm <- samr.norm.data(data$x)
```

Description

Makes the Q-Q plot for a SAM analysis

Usage

```r
samr.plot(samr.obj, del, min.foldchange=0)
```

Arguments

- `samr.obj`: Object returned from call to samr
- `del`: Value of delta to use. Delta is the vertical distance from the 45 degree line to the upper and lower parallel lines that define the SAM threshold rule.
- `min.foldchange`: The minimum fold change desired; should be >1; default is zero, meaning no fold change criterion is applied

Details

Creates the Q-Q plot for a SAM analysis, marking features (genes) that are significant, i.e. lie outside a slab around the 45 degree line of width delta. A gene must also pass the min.foldchange criterion to be called significant, if this criterion is specified in the call.

Author(s)

Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

References

Examples

```r
#generate some example data
set.seed(100)
x <- matrix(rnorm(1000*20), ncol = 20)
dd <- sample(1:1000, size = 100)

u <- matrix(2 * rnorm(100), ncol = 10, nrow = 100)

y <- c(rep(1, 10), rep(2, 10))
data <- list(x = x, y = y, geneid = as.character(1:nrow(x)),
genenames = paste("g", as.character(1:nrow(x)), sep = ""),
logged = TRUE)

samr.obj <- samr(data, resp.type = "Two class unpaired", nperms = 50)
samr.plot(samr.obj, del = .3)
```

Description

Report estimated p-values for each gene, from set of permutations in a SAM analysis

Usage

```r
samr.pvalues.fromperms(tt, ttstar)
```

Arguments

- `tt`: Vector of gene scores, returned by `samr` in component `tt`
- `ttstar`: Matrix of gene scores (p by nperm) from nperm permutations. Returned by `samr` in component `ttstar`

Author(s)

Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

References

Examples
# generate some example data
set.seed(100)
x <- matrix(rnorm(1000*20), ncol=20)
dd <- sample(1:1000, size=100)

u <- matrix(2*runif(100), ncol=10, nrow=100)

y <- c(rep(1, 10), rep(2, 10))
data <- list(x=x, y=y, geneid=as.character(1:nrow(x)),
genenames=paste("g", as.character(1:nrow(x)), sep=""), logged=TRUE)
samr.obj <- samr(data, resp.type="Two class unpaired", nperms=100)

pv <- samr.pvalues.from.perms(samr.obj$t1, samr.obj$t1star)

samr.tail.strength

Estimate tail strength for a dataset, from a SAM analysis

Description
Estimate tail strength for a dataset, from a SAM analysis

Usage
samr.tail.strength(samr.obj)

Arguments
samr.obj  Object returned by samr

Value
A list with components
ts  Estimated tail strength. A number less than or equal to 1. Zero means all genes are null; 1 means all genes are differentially expressed.

se.ts  Estimated standard error of tail strength.

Author(s)
Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

References
# Examples

```r
# generate some example data
data(x, y, censoring = NULL)

# Set seed for reproducibility
set.seed(100)

# Generate features (genes)
x <- matrix(rnorm(1000*20), ncol=20)
dd <- sample(1:1000, size=100)

# Generate outcomes
v <- matrix(2+rnorm(100), ncol=10, nrow=100)
x <- x[dd,1:20] <- x[dd,1:20] + v

# Generate survival times
y <- c(rep(1, 10), rep(2, 10))

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

samr.obj <- samr(data,
          resp.type = "Two class unpaired",
          nperms = 100)

samr.tail.strength(samr.obj)
```

---

**SAMseq**  
*Significance analysis of sequencing data - simple user interface*

**Description**

Correlates a large number of features (e.g., genes) with an outcome variable, such as a group indicator, quantitative variable or survival time. This is a simple user interface for the samr function applied to sequencing data. For array data applications, see the function SAM.

**Usage**

```r
SAMseq(x, y, censoring = NULL,
        resp.type = c("Quantitative", "Two class unpaired",
                      "Survival", "Multiclass", "Two class paired"),
        geneid = NULL, genenames = NULL, nperms = 100,
        random.seed = NULL, nresamp = 20, fdr.output = 0.20)
```

**Arguments**

- **x**: Feature matrix: \( p \) (number of features) by \( n \) (number of samples), one observation per column (missing values allowed)
- **y**: \( n \)-vector of outcome measurements
- **censoring**: \( n \)-vector of censoring censoring.status (1=died or event occurred, 0=survived, or event was censored), needed for a censored survival outcome
- **resp.type**: Problem type: "Quantitative" for a continuous parameter; "Two class unpaired" for two classes with unpaired observations; "Survival" for censored survival outcome; "Multiclass": more than 2 groups; "Two class paired" for two classes with paired observations.
geneid Optional character vector of geneids for output.
genenames Optional character vector of genenames for output.
nperms Number of permutations used to estimate false discovery rates
random.seed Optional initial seed for random number generator (integer)
nresamp Number of resamples used to construct test statistic. Default 20.
fdr.output (Approximate) False Discovery Rate cutoff for output in significant genes table

Details
This is a simple, user-friendly interface to the samr package used on sequencing data. It automatically disables arguments/features that do not apply to sequencing data. It calls samr, samr.compute.delta.table and samr.compute.siggenes.table. samr detects differential expression for microarray data, and sequencing data, and other data with a large number of features. samr is the R package that is called by the "official" SAM Excel Addin. The format of the response vector y and the calling sequence is illustrated in the examples below. A more complete description is given in the SAM manual at http://www-stat.stanford.edu/~tibs/SAM

Value
A list with components

samr.obj Output of samr. See documentation for samr for details
siggenes.table Table of significant genes, output of samr.compute.siggenes.table. This has components: genes.up—matrix of significant genes having positive correlation with the outcome and genes.lo—matrix of significant genes having negative correlation with the outcome. For survival data, genes.up are those genes having positive correlation with risk that is, increased expression corresponds to higher risk (shorter survival) genes.lo are those whose increased expression corresponds to lower risk (longer survival).
delta.table Output of samr.compute.delta.table.
del Value of delta (distance from 45 degree line in SAM plot) for used for creating delta.table and siggenes.table. Changing the input value fdr.output will change the resulting del.
call The calling sequence

Author(s)
Jun Li and Balasubramanian Narasimhan and Robert Tibshirani

References
Examples

###### two class unpaired comparison
set.seed(100)
mu <- matrix(100, 1000, 20)
u[1:100, 1:20] <- 200
mu <- scale(mu, center=FALSE, scale=runif(20, 0.5, 1.5))
x <- matrix(rpois(length(mu), mu), 1000, 20)
y <- c(rep(1:10, 10), rep(2, 10))
samfit <- SAMseq(x, y, resp.type = "Two class unpaired")

# examine significant gene list
print(samfit)

# plot results
plot(samfit)

###### two class paired comparison
set.seed(100)
mu <- matrix(100, 1000, 20)
u[1:100, 1:20] <- 200
mu <- scale(mu, center=FALSE, scale=runif(20, 0.5, 1.5))
x <- matrix(rpois(length(mu), mu), 1000, 20)
y <- c(-(1:10), 1:10)
samfit <- SAMseq(x, y, resp.type = "Two class paired")

# examine significant gene list
print(samfit)

# plot results
plot(samfit)

###### Multiclass comparison
set.seed(100)
mu <- matrix(100, 1000, 20)
u[1:100, 1:10] <- 120
mu[21:50, 1:5] <- 80
mu[51:70, 1:15] <- 150
mu[71:100, 16:20] <- 60
mu <- scale(mu, center=FALSE, scale=runif(20, 0.5, 1.5))
x <- matrix(rpois(length(mu), mu), 1000, 20)
y <- c(rep(1:4, rep(5, 4)))
samfit <- SAMseq(x, y, resp.type = "Multiclass")

# examine significant gene list
print(samfit)

# plot results
plot(samfit)
### Quantitative comparison

```r
set.seed(100)
mu <- matrix(100, 1000, 20)
y <- runif(20, 1, 3)
mu[1:100, ] <- matrix(rep(100 * y, 100), ncol=20, byrow=TRUE)
mu <- scale(mu, center=FALSE, scale=runif(20, 0.5, 1.5))
x <- matrix(rpois(length(mu), mu), 1000, 20)
samfit <- samseq(x, y, resp.type = "Quantitative")

# examine significant gene list
print(samfit)

# plot results
plot(samfit)
```

### Survival comparison

```r
set.seed(100)
mu <- matrix(100, 1000, 20)
y <- runif(20, 1, 3)
mu[1:100, ] <- matrix(rep(100 * y, 100), ncol=20, byrow=TRUE)
mu <- scale(mu, center=FALSE, scale=runif(20, 0.5, 1.5))
x <- matrix(rpois(length(mu), mu), 1000, 20)
y <- y + runif(20, -0.5, 0.5)
censoring.status <- as.numeric(y < 2.3)
y[y >= 2.3] <- 2.3
samfit <- SAMseq(x, y, censoring.status = censoring.status, resp.type = "Survival")

# examine significant gene list
print(samfit)

# plot results
plot(samfit)
```
Index

*Topic nonparametric

SAM, 2
samr, 7
samr.assess.samplesize, 15
samr.assess.samplesize.plot, 17
samr.compute.delta.table, 18
samr.compute.siggenes.table, 19
samr.missrate, 22
samr.plot, 24
samr.pvalues.from.perms, 25
samr.tail.strength, 26
SAMseq, 27

*Topic survival

SAM, 2
samr, 7
samr.assess.samplesize, 15
samr.assess.samplesize.plot, 17
samr.compute.delta.table, 18
samr.compute.siggenes.table, 19
samr.missrate, 22
samr.plot, 24
samr.pvalues.from.perms, 25
samr.tail.strength, 26
SAMseq, 27

*Topic ts

SAM, 2
samr, 7
samr.assess.samplesize, 15
samr.assess.samplesize.plot, 17
samr.compute.delta.table, 18
samr.compute.siggenes.table, 19
samr.missrate, 22
samr.plot, 24
SAMseq, 27

*Topic univar

SAM, 2
samr, 7
samr.assess.samplesize, 15
samr.assess.samplesize.plot, 17
samr.compute.delta.table, 18
samr.compute.siggenes.table, 19
samr.missrate, 22
samr.plot, 24
SAMseq, 27

runSAM, 2

SAM, 2
samr, 7
samr.assess.samplesize, 15
samr.assess.samplesize.plot, 17
samr.compute.delta.table, 18
samr.compute.siggenes.table, 19
samr.estimate.depth, 21
samr.missrate, 22
samr.norm.data, 23
samr.plot, 24
samr.pvalues.from.perms, 25
samr.tail.strength, 26
SAMseq, 27