Package 'ASSIGN'

December 1, 2025

Type Package

Title Adaptive Signature Selection and InteGratioN (ASSIGN)

Version 1.46.0

Date 2014-10-30

Author Ying Shen, Andrea H. Bild, W. Evan Johnson, and Mumtehena Rahman

Maintainer Ying Shen <yshen3@bu.edu>, W. Evan Johnson <wej@bu.edu>, David Jenkins <dfj@bu.edu>, Mumtehena Rahman <moom.rahman@utah.edu>

Depends R (>= 3.4)

Description ASSIGN is a computational tool to evaluate the pathway deregulation/activation status in individual patient samples. ASSIGN employs a flexible Bayesian factor analysis approach that adapts predetermined pathway signatures derived either from knowledge-based literature or from perturbation experiments to the cell-/tissue-specific pathway signatures. The deregulation/activation level of each context-specific pathway is quantified to a score, which represents the extent to which a patient sample encompasses the pathway deregulation/activation signature.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports gplots, graphics, grDevices, msm, Rlab, stats, sva, utils, ggplot2, yaml

biocViews Software, GeneExpression, Pathways, Bayesian

Suggests testthat, BiocStyle, lintr, knitr, rmarkdown

RoxygenNote 7.1.0

VignetteBuilder knitr

URL https://compbiomed.github.io/ASSIGN/

BugReports https://github.com/compbiomed/ASSIGN/issues

git_url https://git.bioconductor.org/packages/ASSIGN

git_branch RELEASE_3_22

git_last_commit 09ba421

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-12-01

Contents

assign.convergence 2
assign.cv.output 4
assign.mcmc 5
assign.output 8
assign.preprocess 9
assign.summary 11
assign.wrapper 13
ComBat.step2 16
excludegenes 17
gather_assign_results 17
geneList1 18
gfrn_geneList 18
merge_drop 18
optimizeGFRN 19
pcaplot 21
runassignGFRN 21
testData1 23
trainingData1 23

Index **24**

`assign.convergence` *Check the convergence of the MCMC chain*

Description

The assign.convergence checks the convergence of the MCMC chain of the model parameters generated by the Gibbs sampling algorithm.

Usage

```
assign.convergence(
  test,
  burn_in = 0,
  iter = 2000,
  parameter = c("B", "S", "Delta", "beta", "kappa", "gamma", "sigma"),
  whichGene,
  whichSample,
  whichPath
)
```

Arguments

test	The list object returned from the assign.mcmc function. The list components are the MCMC chains of the B, S, Delta, beta, gamma, and sigma.
burn_in	The number of burn-in iterations. These iterations are discarded when computing the posterior means of the model parameters. The default is 0.
iter	The number of total iterations. The default is 2000.
parameter	A character string indicating which model parameter is to be checked for convergence. This must be one of "B", "S", "Delta", "beta", "kappa", "gamma", and "sigma".
whichGene	A numerical value indicating which gene is to be checked for convergence. The value has to be in the range between 1 and G.
whichSample	A numerical value indicating which test sample is to be checked for convergence. The value has to be in the range between 1 and N.
whichPath	A numerical value indicating which pathway is to be checked for convergence. The value has to be in the range between 1 and K.

Details

To compute the convergence of the gth gene in B, set whichGene=g, whichSample=NA, whichPath=NA.

To compute the convergence of the gth gene in the kth pathway within the signature matrix (S), set whichGene=g, whichSample=NA, whichPath=NA.

To compute the convergence of the kth pathway in the jth test sample within the pathway activation matrix (A), set whichGene=NA, whichSample=n, whichPath=k.

Value

The assign.convergence function returns the a vector of the estimated values from each Gibbs sampling iteration of the model parameter to be checked, and a trace plot of this parameter.

Author(s)

Ying Shen

Examples

```
## Not run:
# check the 10th gene in the 1st pathway for the convergence
trace.plot <- assign.convergence(test=mcmc.chain, burn_in=0, iter=2000,
                                 parameter="S", whichGene=10, whichSample=NA,
                                 whichPath=1)

## End(Not run)
```

`assign.cv.output` *Cross validation output*

Description

The assign.cv.output function outputs the summary results and plots for the cross validation done on the training dataset.

Usage

```
assign.cv.output(
  processed.data,
  mcmc.pos.mean.trainingData,
  trainingData,
  trainingLabel,
  adaptive_B = FALSE,
  adaptive_S = FALSE,
  mixture_beta = TRUE,
  outputDir
)
```

Arguments

`processed.data`	The list object returned from the assign.preprocess function.
`mcmc.pos.mean.trainingData`	The list object returned from the assign.mcmc function. Notice that for cross validation, the Y argument in the assign.mcmc function should be set as the training dataset.
`trainingData`	The genomic measure matrix of training samples (i.e., gene expression matrix). The dimension of this matrix is probe number x sample number. The default is NULL.
`trainingLabel`	The list linking the index of each training sample to a specific group it belongs to.
`adaptive_B`	Logicals. If TRUE, the model adapts the baseline/background (B) of genomic measures for the test samples. The default is FALSE.
`adaptive_S`	Logicals. If TRUE, the model adapts the signatures (S) of genomic measures for the test samples. The default is FALSE.
`mixture_beta`	Logicals. If TRUE, elements of the pathway activation matrix are modeled by a spike-and-slab mixture distribution. The default is TRUE.
`outputDir`	The path to the directory to save the output files. The path needs to be quoted in double quotation marks.

Details

The assign.cv.output function is suggested to run after the assign.preprocess, assign.mcmc and assign.summary function. For the cross validation, The Y argument in the assign.mcmc function is the output value "trainingData_sub" from the assign.preprocess function.

Value

The assign.cv.output returns one .csv file containing one/multiple pathway activity for each individual training samples, scatter plots of pathway activity for each individual pathway in all the training samples, and heatmap plots for the gene expression signatures for each individual pathways.

Author(s)

Ying Shen

Examples

```
assign.cv.output(processed.data=processed.data,
                 mcmc.pos.mean.trainingData=mcmc.pos.mean,
                 trainingData=trainingData1,
                 trainingLabel=trainingLabel1,
                 adaptive_B=FALSE, adaptive_S=FALSE,
                 mixture_beta=TRUE, outputDir=tempdir)
```

---

assign.mcmc *The Gibbs sampling algorithm to approximate the joint distribution of the model parameters*

Description

The assign.mcmc function uses a Bayesian sparse factor analysis model to estimate the adaptive baseline/background, adaptive pathway signature, and pathway activation status of individual test (disease) samples.

Usage

```
assign.mcmc(
  Y,
  Bg,
  X,
  Delta_prior_p,
  iter = 2000,
  adaptive_B = TRUE,
  adaptive_S = FALSE,
  mixture_beta = TRUE,
  sigma_sZero = 0.01,
  sigma_sNonZero = 1,
  p_beta = 0.01,
  sigma_bZero = 0.01,
  sigma_bNonZero = 1,
  alpha_tau = 1,
  beta_tau = 0.01,
  Bg_zeroPrior = TRUE,
  S_zeroPrior = FALSE,
  ECM = FALSE,
  progress_bar = TRUE
)
```

Arguments

Y	The G x J matrix of genomic measures (i.e., gene expression) of test samples. Y is the testData_sub variable returned from the data.process function. Genes/probes present in at least one pathway signature are retained.
Bg	The G x 1 vector of genomic measures of the baseline/background (B). Bg is the B_vector variable returned from the data.process function. Bg is the starting value of baseline/background level in the MCMC chain.
X	The G x K matrix of genomic measures of the signature. X is the S_matrix variable returned from the data.process function. X is the starting value of pathway signatures in the MCMC chain.
Delta_prior_p	The G x K matrix of prior probability of a gene being "significant" in its associated pathway. Delta_prior_p is the Pi_matrix variable returned from the data.process function.
iter	The number of iterations in the MCMC. The default is 2000.
adaptive_B	Logicals. If TRUE, the model adapts the baseline/background (B) of genomic measures for the test samples. The default is TRUE.
adaptive_S	Logicals. If TRUE, the model adapts the signatures (S) of genomic measures for the test samples. The default is FALSE.
mixture_beta	Logicals. If TRUE, elements of the pathway activation matrix are modeled by a spike-and-slab mixture distribution. The default is TRUE.
sigma_sZero	Each element of the signature matrix (S) is modeled by a spike-and-slab mixture distribution. Sigma_sZero is the variance of the spike normal distribution. The default is 0.01.
sigma_sNonZero	Each element of the signature matrix (S) is modeled by a spike-and-slab mixture distribution. Sigma_sNonZero is the variance of the slab normal distribution. The default is 1.
p_beta	p_beta is the prior probability of a pathway being activated in individual test samples. The default is 0.01.
sigma_bZero	Each element of the pathway activation matrix (A) is modeled by a spike-and-slab mixture distribution. sigma_bZero is the variance of the spike normal distribution. The default is 0.01.
sigma_bNonZero	Each element of the pathway activation matrix (A) is modeled by a spike-and-slab mixture distribution. sigma_bNonZero is the variance of the slab normal distribution. The default is 1.
alpha_tau	The shape parameter of the precision (inverse of the variance) of a gene. The default is 1.
beta_tau	The rate parameter of the precision (inverse of the variance) of a gene. The default is 0.01.
Bg_zeroPrior	Logicals. If TRUE, the prior distribution of baseline/background level follows a normal distribution with mean zero. The default is TRUE.
S_zeroPrior	Logicals. If TRUE, the prior distribution of signature follows a normal distribution with mean zero. The default is TRUE.
ECM	Logicals. If TRUE, ECM algorithm, rather than Gibbs sampling, is applied to approximate the model parameters. The default is FALSE.
progress_bar	Display a progress bar for MCMC. Default is TRUE.

Details

The assign.mcmc function can be set as following major modes. The combination of logical values of adaptive_B, adaptive_S and mixture_beta can form different modes.

Mode A: adaptive_B = FALSE, adaptive_S = FALSE, mixture_beta = FALSE. This is a regression mode without adaptation of baseline/background, signature, and no shrinkage of the pathway activation level.

Mode B: adaptive_B = TRUE, adaptive_S = FALSE, mixture_beta = FALSE. This is a regression mode with adaptation of baseline/background, but without signature, and with no shrinkage of the pathway activation level.

Mode C: adaptive_B = TRUE, adaptive_S = FALSE, mixture_beta = TRUE. This is a regression mode with adaptation of baseline/background, but without signature, and with shrinkage of the pathway activation level when it is not significantly activated.

Mode D: adaptive_B = TRUE, adaptive_S = TRUE, mixture_beta = TRUE. This is a Bayesian factor analysis mode with adaptation of baseline/background, adaptation signature, and with shrinkage of the pathway activation level.

Value

`beta_mcmc`	The iter x K x J array of the pathway activation level estimated in every iteration of MCMC.
`tau2_mcmc`	The iter x G matrix of the precision of genes estimated in every iteration of MCMC
`gamma_mcmc`	The iter x K x J array of probability of pathway being activated estimated in every iteration of MCMC.
`kappa_mcmc`	The iter x K x J array of pathway activation level (adjusted beta scaling between 0 and 1) estimated in every iteration of MCMC.)
`S_mcmc`	The iter x G x K array of signature estimated in every iteration of MCMC.
`Delta_mcmc`	The iter x G x K array of binary indicator of a gene being significant estimated in every iteration of MCMC.

Author(s)

Ying Shen

Examples

```
mcmc.chain <- assign.mcmc(Y=processed.data$testData_sub,
                          Bg = processed.data$B_vector,
                          X=processed.data$S_matrix,
                          Delta_prior_p = processed.data$Pi_matrix,
                          iter = 20, adaptive_B=TRUE, adaptive_S=FALSE,
                          mixture_beta=TRUE)
```

`assign.output` *Prediction/validation output for test data*

Description

The assign.output function outputs the summary results and plots for prediction/validation for the test dataset.

Usage

```
assign.output(
  processed.data,
  mcmc.pos.mean.testData,
  trainingData,
  testData,
  trainingLabel,
  testLabel,
  geneList,
  adaptive_B = TRUE,
  adaptive_S = FALSE,
  mixture_beta = TRUE,
  outputDir
)
```

Arguments

`processed.data`	The list object returned from the assign.preprocess function.
`mcmc.pos.mean.testData`	The list object returned from the assign.mcmc function. Notice that for prediction/validation in the test dataset, the Y argument in the assign.mcmc function should be set as the test dataset.
`trainingData`	The genomic measure matrix of training samples (i.e., gene expression matrix). The dimension of this matrix is probe number x sample number.
`testData`	The genomic measure matrix of test samples (i.e., gene expression matrix). The dimension of this matrix is probe number x sample number.
`trainingLabel`	The list linking the index of each training sample to a specific group it belongs to.
`testLabel`	The vector of the phenotypes/labels of the test samples.
`geneList`	The list that collects the signature genes of one/multiple pathways. Every component of this list contains the signature genes associated with one pathway.
`adaptive_B`	Logicals. If TRUE, the model adapts the baseline/background (B) of genomic measures for the test samples. The default is TRUE.
`adaptive_S`	Logicals. If TRUE, the model adapts the signatures (S) of genomic measures for the test samples. The default is FALSE.
`mixture_beta`	Logicals. If TRUE, elements of the pathway activation matrix are modeled by a spike-and-slab mixture distribution. The default is TRUE.
`outputDir`	The path to the directory to save the output files. The path needs to be quoted in double quotation marks.

Details

The assign.output function is suggested to run after the assign.preprocess, assign.mcmc and assign.summary functions. For the prediction/validation in the test dataset, The Y argument in the assign.mcmc function is the output value "testData_sub" from the assign.preprocess function.

Value

The assign.output returns one .csv file containing one/multiple pathway activity for each individual test samples, scatter plots of pathway activity for each individual pathway in all the test samples, and heatmap plots for the gene expression of the prior signature and posterior signatures (if adaptive_S equals TRUE) of each individual pathway in the test samples.

Author(s)

Ying Shen

Examples

```
assign.output(processed.data = processed.data,
              mcmc.pos.mean.testData = mcmc.pos.mean,
              trainingData = trainingData1, testData = testData1,
              trainingLabel = trainingLabel1, testLabel = testLabel1,
              geneList = NULL, adaptive_B = TRUE, adaptive_S = FALSE,
              mixture_beta = TRUE, outputDir = tempdir)
```

---

assign.preprocess *Input data preprocessing*

Description

The assign.preprocess function is used to perform quality control on the user-provided input data and generate starting values and/or prior values for the model parameters. The assign.preprocess function is optional. For users who already have the correct format for the input of the assign function, they can skip this step and go directly to the assign.mcmc function.

Usage

```
assign.preprocess(
  trainingData = NULL,
  testData,
  anchorGenes = NULL,
  excludeGenes = NULL,
  trainingLabel,
  geneList = NULL,
  n_sigGene = NA,
  theta0 = 0.05,
  theta1 = 0.9,
  pctUp = 0.5,
  geneselect_iter = 500,
  geneselect_burn_in = 100,
  progress_bar = TRUE
)
```

Arguments

trainingData	The genomic measure matrix of training samples (i.e., gene expression matrix). The dimension of this matrix is probe number x sample number. The default is NULL.
testData	The genomic measure matrix of test samples (i.e., gene expression matrix). The dimension of this matrix is probe number x sample number.
anchorGenes	A list of genes that will be included in the signature even if they are not chosen during gene selection.
excludeGenes	A list of genes that will be excluded from the signature even if they are chosen during gene selection.
trainingLabel	The list linking the index of each training sample to a specific group it belongs to. See details and examples for more information.
geneList	The list that collects the signature genes of one/multiple pathways. Every component of this list contains the signature genes associated with one pathway. The default is NULL.
n_sigGene	The vector of the signature genes to be identified for one pathway. n_sigGene needs to be specified when geneList is set NULL. The default is NA. See examples for more information.
theta0	The prior probability for a gene to be significant, given that the gene is NOT defined as "significant" in the signature gene lists provided by the user. The default is 0.05.
theta1	The prior probability for a gene to be significant, given that the gene is defined as "significant" in the signature gene lists provided by the user. The default is 0.9.
pctUp	By default, ASSIGN bayesian gene selection chooses the signature genes with an equal fraction of genes that increase with pathway activity and genes that decrease with pathway activity. Use the pctUp parameter to modify this fraction. Set pctUP to NULL to select the most significant genes, regardless of direction. The default is 0.5
geneselect_iter	The number of iterations for bayesian gene selection. The default is 500.
geneselect_burn_in	The number of burn-in iterations for bayesian gene selection. The default is 100
progress_bar	Display a progress bar for gene selection. Default is TRUE.

Details

The assign.preprocess is applied to perform quality control on the user-provided genomic data and meta data, re-format the data in a way that can be used in the following analysis, and generate starting/prior values for the pathway signature matrix. The output values of the assign.preprocess function will be used as input values for the assign.mcmc function.

For training data with 1 control group and 3 experimental groups (10 samples/group; all 3 experimental groups share 1 control group), the trainingLabel can be specified as: trainingLabel <- list(control = list(expr1=1:10, expr2=1:10, expr3=1:10), expr1 = 11:20, expr2 = 21:30, expr3 = 31:40)

For training data with 3 control groups and 3 experimental groups (10 samples/group; Each experimental group has its corresponding control group), the trainingLabel can be specified as: trainingLabel <- list(control = list(expr1=1:10, expr2=21:30, expr3=41:50), expr1 = 11:20, expr2 = 31:40, expr3 = 51:60)

It is highly recommended that the user use the same experiment name when specifying control indices and experimental indices.

Value

`trainingData_sub`
: The G x N matrix of G genomic measures (i.e., gene expression) of N training samples. Genes/probes present in at least one pathway signature are retained. Only returned when the training dataset is available.

`testData_sub`
: The G x N matrix of G genomic measures (i.e., gene expression) of N test samples. Genes/probes present in at least one pathway signature are retained.

`B_vector`
: The G x 1 vector of genomic measures of the baseline/background. Each element of the B_vector is calculated as the mean of the genomic measures of the control samples in training data.

`S_matrix`
: The G x K matrix of genomic measures of the signature. Each column of the S_matrix represents a pathway. Each element of the S_matrix is calculated as the mean of genomic measures of the experimental samples minus the mean of the control samples in the training data.

`Delta_matrix`
: The G x K matrix of binary indicators. Each column of the Delta_matrix represents a pathway. The elements in Delta_matrix are binary (0, insignificant gene; 1, significant gene).

`Pi_matrix`
: The G x K matrix of probability p of a Bernoulli distribution. Each column of the Pi_matrix represents a pathway. Each element in the Pi_matrix is the probability of a gene to be significant in its associated pathway.

`diffGeneList`
: The list that collects the signature genes of one/multiple pathways generated from the training samples or from the user provided gene list. Every component of this list contains the signature genes associated with one pathway.

Author(s)

Ying Shen

Examples

```
processed.data <- assign.preprocess(trainingData=trainingData1,
                                    testData=testData1,
                                    trainingLabel=trainingLabel1,
                                    geneList=geneList1)
```

---

assign.summary *Summary of the model parameters estimated by the Gibbs sampling algorithm*

Description

The assign.summary function computes the posterior mean of the model parameters estimated in every iteration during the Gibbs sampling.

Usage

```
assign.summary(
  test,
  burn_in = 1000,
  iter = 2000,
  adaptive_B = TRUE,
  adaptive_S = FALSE,
  mixture_beta = TRUE
)
```

Arguments

test	The list object returned from the assign.mcmc function. The list components are the MCMC chains of the B, S, Delta, beta, gamma, and sigma.
burn_in	The number of burn-in iterations. These iterations are discarded when computing the posterior means of the model parameters. The default is 1000.
iter	The number of total iterations. The default is 2000.
adaptive_B	Logicals. If TRUE, the model adapts the baseline/background (B) of genomic measures for the test samples. The default is TRUE.
adaptive_S	Logicals. If TRUE, the model adapts the signatures (S) of genomic measures for the test samples. The default is FALSE.
mixture_beta	Logicals. If TRUE, elements of the pathway activation matrix are modeled by a spike-and-slab mixture distribution. The default is TRUE.

Details

The assign.summary function is suggested to run after the assign.convergence function, which is used to check the convergence of the MCMC chain. If the MCMC chain does not converge to a stationary phase, more iterations are required in the assign.mcmc function. The number of burn-in iterations is usually set to be half of the number of total iterations, meaning that the first half of the MCMC chain is discarded when computing the posterior means.

Value

beta_pos	The N x K matrix of the posterior mean of the pathway activation level in test samples (transposed matrix A). Columns:K pathways; rows: N test samples
sigma_pos	The G x 1 vector of the posterior mean of the variance of gene.
kappa_pos	The N x K matrix of posterior mean of pathway activation level in test samples (transposed matrix A) (adjusted beta_pos scaling between 0 and 1). Columns:K pathways; rows: N test samples
gamma_pos	The N x K matrix of the posterior probability of pathways being activated in test samples.
S_pos	The G x K matrix of the posterior mean of pathway signature genes.
Delta_pos	The G x K matrix of the posterior probability of genes being significant in the associated pathways.

Author(s)

Ying Shen

Examples

```
data(trainingData1)
data(testData1)
data(geneList1)
trainingLabel1 <- list(control = list(bcat=1:10, e2f3=1:10, myc=1:10,
                                      ras=1:10, src=1:10),
                       bcat = 11:19, e2f3 = 20:28, myc= 29:38,
                       ras = 39:48, src = 49:55)

processed.data <- assign.preprocess(trainingData=trainingData1,
testData=testData1, trainingLabel=trainingLabel1, geneList=geneList1)
mcmc.chain <- assign.mcmc(Y=processed.data$testData_sub,
                          Bg = processed.data$B_vector,
                          X=processed.data$S_matrix,
                          Delta_prior_p = processed.data$Pi_matrix,
                          iter = 20, adaptive_B=TRUE, adaptive_S=FALSE,
                          mixture_beta=TRUE)
mcmc.pos.mean <- assign.summary(test=mcmc.chain, burn_in=10, iter=20,
                                adaptive_B=TRUE, adaptive_S=FALSE,
                                mixture_beta = TRUE)
```

`assign.wrapper` *ASSIGN All-in-one function*

Description

The assign.wrapper function integrates the assign.preprocess, assign.mcmc, assign.summary, assign.output, assign.cv.output functions into one wrapper function.

Usage

```
assign.wrapper(
  trainingData = NULL,
  testData,
  trainingLabel,
  testLabel = NULL,
  geneList = NULL,
  anchorGenes = NULL,
  excludeGenes = NULL,
  n_sigGene = NA,
  adaptive_B = TRUE,
  adaptive_S = FALSE,
  mixture_beta = TRUE,
  outputDir,
  p_beta = 0.01,
  theta0 = 0.05,
  theta1 = 0.9,
  iter = 2000,
  burn_in = 1000,
  sigma_sZero = 0.01,
  sigma_sNonZero = 1,
```

```
  S_zeroPrior = FALSE,
  pctUp = 0.5,
  geneselect_iter = 500,
  geneselect_burn_in = 100,
  outputSignature_convergence = FALSE,
  ECM = FALSE,
  progress_bar = TRUE,
  override_S_matrix = NULL
)
```

Arguments

trainingData	The genomic measure matrix of training samples (i.e., gene expression matrix). The dimension of this matrix is probe number x sample number. The default is NULL.
testData	The genomic measure matrix of test samples (i.e., gene expression matrix). The dimension of this matrix is probe number x sample number.
trainingLabel	The list linking the index of each training sample to a specific group it belongs to. See examples for more information.
testLabel	The vector of the phenotypes/labels of the test samples. The default is NULL.
geneList	The list that collects the signature genes of one/multiple pathways. Every component of this list contains the signature genes associated with one pathway. The default is NULL.
anchorGenes	A list of genes that will be included in the signature even if they are not chosen during gene selection.
excludeGenes	A list of genes that will be excluded from the signature even if they are chosen during gene selection.
n_sigGene	The vector of the signature genes to be identified for one pathway. n_sigGene needs to be specified when geneList is set NULL. The default is NA. See examples for more information.
adaptive_B	Logicals. If TRUE, the model adapts the baseline/background (B) of genomic measures for the test samples. The default is TRUE.
adaptive_S	Logicals. If TRUE, the model adapts the signatures (S) of genomic measures for the test samples. The default is FALSE.
mixture_beta	Logicals. If TRUE, elements of the pathway activation matrix are modeled by a spike-and-slab mixture distribution. The default is TRUE.
outputDir	The path to the directory to save the output files. The path needs to be quoted in double quotation marks.
p_beta	p_beta is the prior probability of a pathway being activated in individual test samples. The default is 0.01.
theta0	The prior probability for a gene to be significant, given that the gene is NOT defined as "significant" in the signature gene lists provided by the user. The default is 0.05.
theta1	The prior probability for a gene to be significant, given that the gene is defined as "significant" in the signature gene lists provided by the user. The default is 0.9.
iter	The number of iterations in the MCMC. The default is 2000.

`burn_in` The number of burn-in iterations. These iterations are discarded when computing the posterior means of the model parameters. The default is 1000.

`sigma_sZero` Each element of the signature matrix (S) is modeled by a spike-and-slab mixture distribution. Sigma_sZero is the variance of the spike normal distribution. The default is 0.01.

`sigma_sNonZero` Each element of the signature matrix (S) is modeled by a spike-and-slab mixture distribution. Sigma_sNonZero is the variance of the slab normal distribution. The default is 1.

`S_zeroPrior` Logicals. If TRUE, the prior distribution of signature follows a normal distribution with mean zero. The default is TRUE.

`pctUp` By default, ASSIGN bayesian gene selection chooses the signature genes with an equal fraction of genes that increase with pathway activity and genes that decrease with pathway activity. Use the pctUp parameter to modify this fraction. Set pctUP to NULL to select the most significant genes, regardless of direction. The default is 0.5

`geneselect_iter`
The number of iterations for bayesian gene selection. The default is 500.

`geneselect_burn_in`
The number of burn-in iterations for bayesian gene selection. The default is 100

`outputSignature_convergence`
Create a pdf of the MCMC chain. The default is FALSE.

`ECM` Logicals. If TRUE, ECM algorithm, rather than Gibbs sampling, is applied to approximate the model parameters. The default is FALSE.

`progress_bar` Display a progress bar for MCMC and gene selection. Default is TRUE.

`override_S_matrix`
Replace the S_matrix created by assign.preprocess with the matrix provided in override_S_matrix. This can be used to indicate the expected directions of genes in a signature if training data is not provided.

Details

The assign.wrapper function is an all-in-one function which outputs the necessary results for basic users. For users who need more intermediate results for model diagnosis, it is better to run the assign.preprocess, assign.mcmc, assign.convergence, assign.summary functions separately and extract the output values from the returned list objects of those functions.

Value

The assign.wrapper returns one/multiple pathway activity for each individual training sample and test sample, scatter plots of pathway activity for each individual pathway in the training and test data, heatmap plots for gene expression signatures for each individual pathway, heatmap plots for the gene expression of the prior and posterior signatures (if adaptive_S equals TRUE) of each individual pathway in the test data

Author(s)

Ying Shen and W. Evan Johnson

Examples

```
data(trainingData1)
data(testData1)
data(geneList1)

trainingLabel1 <- list(control = list(bcat=1:10, e2f3=1:10, myc=1:10,
                                      ras=1:10, src=1:10),
                       bcat = 11:19, e2f3 = 20:28, myc= 29:38, ras = 39:48,
                       src = 49:55)
testLabel1 <- rep(c("subtypeA","subtypeB"), c(53,58))

assign.wrapper(trainingData=trainingData1, testData=testData1,
               trainingLabel=trainingLabel1, testLabel=testLabel1,
               geneList=geneList1, adaptive_B=TRUE, adaptive_S=FALSE,
               mixture_beta=TRUE, outputDir=tempdir, p_beta=0.01,
               theta0=0.05, theta1=0.9, iter=20, burn_in=10)
```

`ComBat.step2` *Perform the second step of ComBat*

Description

The first ComBat step (on the signatures only) has already been performed. This step performs batch correction on the test data, using reference batch ComBat, to prepare the test data for ASSIGN analysis.

Usage

```
ComBat.step2(
  testData,
  pcaPlots = FALSE,
  combat_train = NULL,
  plots_to_console = FALSE
)
```

Arguments

`testData`	The input test data to batch correct
`pcaPlots`	a logical value indicating whether or not the function should create PCA plots. The default is FALSE.
`combat_train`	the ComBat training data data frame. If you do not have this, the function will attempt to download it from the internet. Please contact the developers if you have any issues with access to the file.
`plots_to_console`	By default this function will write PDF versions of the plots. Set this to TRUE to send the plots to the command line. The default is FALSE.

Details

This function downloads the training data from the internet, so an internet connection is necessary

Value

A list of data.frames is returned, including control (GFP) and signature data, as well as the batch corrected test data. This data can go directly into the runassign.single and runassign.multi functions, or subsetted to go directly into ASSIGN.

`excludegenes` *Exclude Gene List*

Description

Overexpression signatures may contain genes that are consistently differentially expressed. This list was compiled based on the GFRN gene list. These genes appear in at least 60

Format

character vector of commonly differentially expressed genes

Source

Bild et al.

`gather_assign_results` *Gather the ASSIGN results in a specific directory*

Description

Gather the ASSIGN results in a specific directory

Usage

```
gather_assign_results(path = ".")
```

Arguments

`path`	The path to the directory containing ASSIGN results. The default is the current working directory.

Value

A data frame of ASSIGN predictions from each run in the directory

geneList1 *Pathway signature gene sets*

Description

Signature genes for 5 oncogenic pathways.

Format

List with 5 components representing each pathway. 200 signature genes are selected for each pathway.

Source

Bild et al. (2006) Oncogenic pathway signatures in human cancers as a guide to targeted therapies. Nature, 439, 353-357.

gfrn_geneList *Pathway Signature Gene Lists*

Description

Pathway signature gene lists have been optimized based on correlations of pathway activity data and protein data. The gene lists can be used to avoid the bayesian gene selection step of ASSIGN, which will decrease the amount of time it takes to run ASSIGN.

Format

List of gene lists for akt, bad, egfr, her2, igf1r, krasgv, krasqh, and raf

Source

Bild et al.

merge_drop *Combine two data frames*

Description

Combine two data frames

Usage

```
merge_drop(x, y, by = 0, ...)
```

Arguments

x	The first data frame to be coerced to one.
y	The second data frame to be coerced to one.
by	specifications of the columns used for merging. The default is by row names
...	arguments to be passed to or from methods.

Value

The returned data frame is the combination of x and y, with the rownames properly assigned.

Examples

```
## Not run:
merged.df <- merge_drop(df1,df2)

## End(Not run)
```

---

optimizeGFRN *Optimize GFRN gene lists lengths*

Description

This function runs ASSIGN pathway prediction on gene list lengths from 5 to 500 to find the optimum gene list length for the GFRN pathways by correlating the ASSIGN predictions to a matrix of correlation data that you provide. This function takes a long time to run because you are running ASSIGN many times on many pathways, so I recommend parallelizing by pathway or running the ASSIGN predictions first (long and parallelizable) and then running the correlation step (quick) separately.

Usage

```
optimizeGFRN(
  indata,
  correlation,
  correlationList,
  run = c("akt", "bad", "egfr", "her2", "igf1r", "krasgv", "krasqh", "raf"),
  run_ASSIGN_only = FALSE,
  correlation_only = FALSE,
  keep_optimized_only = FALSE,
  pathway_lengths = c(seq(5, 20, 5), seq(25, 275, 25), seq(300, 500, 50)),
  iter = 1e+05,
  burn_in = 50000
)
```

Arguments

indata	The list of data frames from ComBat.step2
correlation	A matrix of data to correlate ASSIGN predictions to. The number of rows should be the same and in the same order as indata

correlationList
: A list that shows which columns of correlation should be used for each pathway. See below for more details

run
: specifies the pathways to predict. The default list will cause all eight pathways to be run in serial. Specify a pathway ("akt", "bad", "egfr", etc.) or list of pathways to run those pathways only.

run_ASSIGN_only
: a logical value indicating if you want to run the ASSIGN predictions only. Use this to parallelize ASSIGN runs across a compute cluster or across compute threads

correlation_only
: a logical value indicating if you want to run the correlation step only. The function will find the ASSIGN runs in the cwd and optimize them based on the correlation data matrix.

keep_optimized_only
: a logical value indicating if you want to keep all of the ASSIGN run results, or only the runs that provided the optimum ASSIGN correlations. This will delete all directories in the current working directory that match the pattern "_gene_list". The default is FALSE

pathway_lengths
: The gene list lengths that should be run. The default is the 20 pathway lengths that were used in the paper, but this list can be customized to which pathway lengths you are willing to accept

iter
: The number of iterations in the MCMC.

burn_in
: The number of burn-in iterations. These iterations are discarded when computing the posterior means of the model parameters.

Value

ASSIGN runs are output to the current workingdirectory. This function returns the correlation data and the optimized gene lists that you can use with runassignGFRN to try these lists on other data.

Examples

```
## Not run:
testData <- read.table(paste0("https://drive.google.com/uc?authuser=0",
                              "&id=1mJICN4z_aCeh4JuPzNfm8GR_lkJOhWFr",
                              "&export=download"),
                       sep='\t', row.names=1, header=1)
corData <- read.table(paste0("https://drive.google.com/uc?authuser=0",
                             "&id=1MDWVP2jBsAAcMNcNFKE74vYl-orpo7WH",
                             "&export=download"),
                      sep='\t', row.names=1, header=1)
corData$negAkt <- -1 * corData$Akt
corData$negPDK1 <- -1 * corData$PDK1
corData$negPDK1p241 <- -1 * corData$PDK1p241

corList <- list(akt=c("Akt","PDK1","PDK1p241"),
                bad=c("negAkt","negPDK1","negPDK1p241"),
                egfr=c("EGFR","EGFRp1068"),
                her2=c("HER2","HER2p1248"),
                igf1r=c("IGFR1","PDK1","PDK1p241"),
                krasgv=c("EGFR","EGFRp1068"),
```

```
                krasqh=c("EGFR","EGFRp1068"),
                raf=c("MEK1","PKCalphap657","PKCalpha"))

combat.data <- ComBat.step2(testData, pcaPlots = TRUE)

optimization_results <- optimizeGFRN(combat.data, corData, corList)

## End(Not run)
```

`pcaplot` *Display a PCA Plot of the Data*

Description

Display a PCA Plot of the Data

Usage

```
pcaplot(mat, sub, center = TRUE, scale = TRUE, plottitle = "PCA")
```

Arguments

`mat`	The data frame on which to perform pca.
`sub`	The number of samples in this batch, from left to right in the data frame
`center`	a logical value indicating whether the variables should be shifted to be zero centered. The default is TRUE
`scale`	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is TRUE
`plottitle`	The title to display above your PCA plot. The default is "PCA".

Value

A PCA plot is displayed

`runassignGFRN` *Run optimized single pathway ASSIGN*

Description

This function runs eight ASSIGN runs based on the pathway optimizations from the paper. You can run all eight pathways in serial, or call this function and specify the run parameter to run a specific pathway. Some ASSIGN parameters can be customized using this function. The default values were used in the analysis for the paper.

Usage

```
runassignGFRN(
  indata,
  run = c("akt", "bad", "egfr", "her2", "igf1r", "krasgv", "raf"),
  optimized_geneList = NULL,
  use_seed = 1234,
  sigma_sZero = 0.05,
  sigma_sNonZero = 0.5,
  S_zeroPrior = FALSE,
  iter = 1e+05,
  burn_in = 50000,
  exclude_common_genes = FALSE,
  adaptive_S = TRUE,
  ECM = FALSE
)
```

Arguments

indata	The list of data frames from ComBat.step2
run	specifies the pathways to predict. The default list will cause all eight pathways to be run in serial. Specify a pathway ("akt", "bad", "egfr", etc.) or list of pathways to run those pathways only.
optimized_geneList	a list of custom optimized gene lists for the gfrn pathways either created manually or output by optimizeGFRN
use_seed	Set the seed before running ASSIGN. This will make the result consistent between runs. The default is 1234. Set use_seed as FALSE to not set a seed.
sigma_sZero	Each element of the signature matrix (S) is modeled by a spike-and-slab mixture distribution. Sigma_sZero is the variance of the spike normal distribution. The default is 0.05.
sigma_sNonZero	Each element of the signature matrix (S) is modeled by a spike-and-slab mixture distribution. Sigma_sNonZero is the variance of the slab normal distribution. The default is 0.5.
S_zeroPrior	Logicals. If TRUE, the prior distribution of signature follows a normal distribution with mean zero. The default is FALSE.
iter	The number of iterations in the MCMC. The default is 100000.
burn_in	The number of burn-in iterations. These iterations are discarded when computing the posterior means of the model parameters. The default is 50000.
exclude_common_genes	Remove commonly differentially expressed genes for overexpression signatures. The default is FALSE.
adaptive_S	Logical. If TRUE, the model adapts the signatures (S) of genomic measures for the test samples. The default for GFRN analysis is TRUE.
ECM	Logicals. If TRUE, ECM algorithm, rather than Gibbs sampling, is applied to approximate the model parameters. The default is FALSE.

Value

Data is output to the current working directory in a results directory.

Examples

```
## Not run:
testData <- read.table(paste0("https://drive.google.com/uc?authuser=0&",
                              "id=1mJICN4z_aCeh4JuPzNfm8GR_lkJOhWFr",
                              "&export=download"),
                       sep='\t', row.names=1, header=1)
combat.data <- ComBat.step2(testData, pcaPlots = TRUE)
runassignGFRN(combat.data)

## End(Not run)
```

`testData1` *Gene expression profiling from cancer patients (test dataset)*

Description

Gene expression datasets for 111 lung cancer patient samples, including 53 cases of lung adenocarcinoma and 58 cases of lung squamous carcinoma.

Format

Data frame with 1000 genes/probes (rows) and 111 samples (columns)

Source

Bild et al. (2006) Oncogenic pathway signatures in human cancers as a guide to targeted therapies. Nature, 439, 353-357.

`trainingData1` *Gene expression profiling from cell line perturbation experiments (training dataset)*

Description

Gene expression datasets for 5 oncogenic pathway perturbation experiments, including B-Catenin, E2F3, MYC, RAS, and SRC pathways.

Format

Data frame with 1000 genes/probes (rows) and 55 samples (columns)

Source

Bild et al. (2006) Oncogenic pathway signatures in human cancers as a guide to targeted therapies. Nature, 439, 353-357.

Index

∗ **datasets**
excludegenes, 17
geneList1, 18
gfrn_geneList, 18
testData1, 23
trainingData1, 23

assign.convergence, 2
assign.cv.output, 4
assign.mcmc, 5
assign.output, 8
assign.preprocess, 9
assign.summary, 11
assign.wrapper, 13

ComBat.step2, 16

excludegenes, 17

gather_assign_results, 17
geneList1, 18
gfrn_geneList, 18

merge_drop, 18

optimizeGFRN, 19

pcaplot, 21

runassignGFRN, 21

testData1, 23
trainingData1, 23