

Package ‘EquiNorm’

February 14, 2012

Type Package

Title Normalize expression data using equivalently expressed genes

Version 2.0

Date 2011-03-24

Author Li-Xuan Qin <qinl@mskcc.org>, Jaya M. Satagopan
<satagopj@mskcc.org>, Brian Denton <dentonb@mskcc.org>

Maintainer Li-Xuan Qin <qinl@mskcc.org>

Description Normalize oligonucleotide gene expression data using equivalently expressed genes in experiments comparing two groups of samples (for example, case-control samples), as outlined in L-X Qin and JM Satagopan (2009).

Depends R (>= 2.8.0), Matrix, lattice

Imports lme4, mvtnorm, lemma

License GPL (>= 2)

LazyLoad yes

LazyData yes

URL <http://www.mskcc.org/mskcc/html/91979.cfm>

Repository CRAN

Date/Publication 2012-02-03 18:15:17

R topics documented:

EquiNorm-package	2
data.simulated	4
data.x	6
data.y	6
equi.gene.norm	6
est.hat	10
lambda.start	11
lemma.maxIts	11
lemma.outdir	12
lemma.plots	12
lemma.tol	12
maxIter	12
pi.start	13
tolerance	13

Index	14
--------------	-----------

EquiNorm-package	<i>Normalize expression data using equivalently expressed genes</i>
------------------	---

Description

Normalize oligonucleotide gene expression data using equivalently expressed genes in experiments comparing two groups of samples (for example, tumor samples vs. normal samples), as outlined in LX Qin and JM Satagopan (2009).

Details

Package:	EquiNorm
Type:	Package
Version:	2.0
Date:	2011-24-03
License:	LGPL >= 2.0
LazyLoad:	yes

Existing methods for data normalization often assume that there are few or symmetric differential expression, but this assumption does not always hold. Alternatively, non-differentially expressed genes may be used for array normalization. However, it is unknown a priori which genes are non-differentially expressed. This procedure implements a hierarchical mixture model framework to simultaneously identify non-differentially expressed genes and normalize arrays using these genes.

The required inputs are an array of gene expression data (`data.y`) and a binary sequence indicating the disease status of each subject (`data.x`). Additional parameters are described in the documentation for the `equi.gene.norm()` function.

Author(s)

Li-Xuan Qin <qinl@mskcc.org>, Jaya M. Satagopan <satagopj@mskcc.org>
 Maintainer: Brian Denton <dentonb@mskcc.org>

References

Qin LX and Satagopan J. Normalization method for transcriptional studies of heterogeneous samples - simultaneous array normalization and identification of equivalent expression. *Statistical Applications in Genetics and Molecular Biology* 2009, 8: Article 10.

Haim Bar and Elizabeth Schifano. (2010). lemma: Laplace approximated EM Microarray Analysis. R package version 1.3-1. <http://CRAN.R-project.org/package=lemma>

Douglas Bates <bates@stat.wisc.edu> and Martin Maechler <maechler@R-project.org> (2010). lme4: Linear mixed-effects models using S4 classes. R package version 0.999375-37. <http://CRAN.R-project.org/package=lme4>

Douglas Bates <bates@stat.wisc.edu> and Martin Maechler <maechler@stat.math.ethz.ch> (2010). Matrix: Sparse and Dense Matrix Classes and Methods. R package version 0.999375-46. <http://CRAN.R-project.org/package=Matrix>

Sarkar, Deepayan (2008) Lattice: Multivariate Data Visualization with R. Springer, New York. ISBN 978-0-387-75968-5

Examples

```
#####
# Load data.simulated data      #
#####

data(data.simulated)

#####
# Run equi.gene.norm function.      #
# data.x and data.y are the only required parameters; #
# all others are optional.          #
#####

est.hat.lemma <- equi.gene.norm( data.simulated$data.y, data.simulated$data.x,
  pi.start = 0.1, lambda.start = 0.9, tolerance = 1E-3, maxIter = 10,
  lemma.outdir = tempdir(), lemma.tol = 1E-6, lemma.maxIts = 50000, lemma.plots = FALSE )

#####
# Print results from estimation and compare with true values.      #
# The true values are known because the example data are data.simulated. #
#####

# alpha.hat is an n-element list containing the estimated array effect for each of the arrays.
# pi.hat is the estimated proportion of genes that are differentially expressed.
```

```

# lambda.hat is the estimated proportion of differentially expressed genes that are over-expressed.
# mu0.hat is the treatment effect of gene expression among over-expressed genes.
# muU.hat is the treatment effect of gene expression among under-expressed genes.
# tau2.hat is the variance of gene effects.
# psi2.hat is the variance of treatment effect for over-expressed genes.
# xi2.hat is the variance of the treatment effect for under-expressed genes.
# sigma2.hat is the variance of measurement error.

est.hat.lemma$theta.hat

# Percent error for parameter estimates
100*(est.hat.lemma$theta.hat$pi.hat - data.simulated$true.values$pi)/data.simulated$true.values$pi
100*(est.hat.lemma$theta.hat$lambda.hat - data.simulated$true.values$lambda)/data.simulated$true.values$lambda
100*(est.hat.lemma$theta.hat$muU.hat - data.simulated$true.values$muU)/data.simulated$true.values$muU
100*(est.hat.lemma$theta.hat$mu0.hat - data.simulated$true.values$mu0)/data.simulated$true.values$mu0
100*(est.hat.lemma$theta.hat$tau2.hat - data.simulated$true.values$tau2)/data.simulated$true.values$tau2
100*(est.hat.lemma$theta.hat$psi2.hat - data.simulated$true.values$psi2)/data.simulated$true.values$psi2
100*(est.hat.lemma$theta.hat$xi2.hat - data.simulated$true.values$xi2)/data.simulated$true.values$xi2
100*(est.hat.lemma$theta.hat$sigma2.hat - data.simulated$true.values$sigma2)/data.simulated$true.values$sigma2

# Percent error for estimated array effects
alpha.compare <- cbind(data.simulated$data.x, data.simulated$true.values$alpha, est.hat.lemma$theta.hat$alpha.hat,
                      100*(est.hat.lemma$theta.hat$alpha.hat - data.simulated$true.values$alpha)/data.simulated$true.values$alpha)
colnames(alpha.compare) <- c("data.simulated$data.x", "True alpha", "alpha.hat", "Percent Error" )
alpha.compare

# Plot of true array effects versus estimated array effects

Control.count <- length( which( data.simulated$data.x == 0 ) )
Case.count <- length( which( data.simulated$data.x == 1 ) )

plot( x = data.simulated$true.values$alpha, y = est.hat.lemma$theta.hat$alpha.hat,
      xlab = "True alpha", ylab = "Estimated alpha",
      main = "Plot of True alpha vs. Estimated alpha",
      col = c(rep("black",Control.count), rep("blue",Case.count) ) )
abline( a = 0, b = 1, col = "red" )
legend( "bottomright", pch = c(1,1), col = c("black", "blue" ),
       legend = c("Control","Case") )

# Cross-tabulations for predictions of under-expression and over-expression

table( data.simulated$true.values$data.u, est.hat.lemma$e.hat$u.hat )
prop.table(table( data.simulated$true.values$data.u, est.hat.lemma$e.hat$u.hat ))

table( data.simulated$true.values$data.o, est.hat.lemma$e.hat$o.hat )
prop.table(table( data.simulated$true.values$data.o, est.hat.lemma$e.hat$o.hat ))

```

Description

A list containing simulated datasets - i.e. an example data.x and data.y.

Format

The format is:

List of 3

\\$ data.y : num [1:S, 1:n, 1:P] 25.2 20.5 27.2 25.1 22.7 ...

\\$ data.x : int [1:n] 0 0 0 0 0 0 0 0 0 ...

\\$ **true.values** : List of 11

\\$ lambda: num 0.2

\\$ pi : num 0.1

\\$ muU : num -2

\\$ muO : num 2

\\$ tau2 : num 6.25

\\$ psi2 : num 1

\\$ xi2 : num 0.562

\\$ sigma2: num 0.09

\\$ alpha : num [1:n] 24.6 22.1 22 21.9 21.6 ...

\\$ data.o: num [1:S] 1 1 1 1 1 1 1 1 1 ...

\\$ data.u: num [1:S] 0 0 0 0 0 0 0 0 0 ...

Details

data.y is a three-dimensional array of simulated expression data from 20 samples, 500 probesets, and 11 probes per probeset.

data.x is a binary vector indicating disease status for each of 20 simulated samples.

true.values is an 11-element list containing the following parameters used to simulate data.y and data.x:

alpha is a 20-element list containing the estimated array effect for each of the arrays.

pi is the estimated proportion of genes are differentially expressed.

lambda is the estimated proportion of differentially expressed genes are over-expressed.

muO is the treatment effect of gene expression among over-expressed genes.

muU is the treatment effect of gene expression among under-expressed genes.

tau2 is the variance of gene effects.

psi2 is the variance of treatment effect for over-expressed genes.

xi2 is the variance of the treatment effect for under-expressed genes.

sigma2 is the variance of measurement error.

data.x	<i>A vector indicating group (e.g. disease status) for each subject.</i>
--------	--

Description

Vector indicating group (e.g. disease status) for each subject. The vector data.x is a binary vector of length n , where n is the number of samples.

Format

```
int [1:n] 0 0 1 0 0 1 1 0 1 0 ...
```

data.y	<i>An array of gene expression data.</i>
--------	--

Description

A 3-dimensional array containing probe-level gene expression data. The array data.y contains values for S probesets, n samples, and P probes.

Format

The format is: num [1:S, 1:n, 1:P] 10.03 6.09 7.27 9.85 7.47 ...

equi.gene.norm	<i>Normalize expression data using equivalently expressed genes</i>
----------------	---

Description

Normalize oligonucleotide gene expression data using equivalently expressed genes in experiments comparing two groups of samples (for example, tumor samples vs. normal samples), as outlined in LX Qin and JM Satagopan (2009).

Usage

```
equi.gene.norm( data.y, data.x,
                pi.start = 0.1, lambda.start = 0.9, tolerance = 1E-3, maxIter = 10,
                lemma.outdir = tempdir(), lemma.tol = 1E-6, lemma.maxIts = 50000, lemma.plots = FALSE )
```

Arguments

<code>data.y</code>	Array of observed probe-level gene expressions. This is a 3-dimensional array containing probe-level gene expression data. The array <code>data.y</code> contains values for S probesets, n samples, and P probes
<code>data.x</code>	Vector indicating sample group (e.g. disease status with 0 for Control, 1 for Case). The vector <code>data.x</code> is a binary vector of length n , where n is the number of samples.
<code>pi.start</code>	(Optional) Starting value for the proportion of differentially expressed genes. The default value is 0.1.
<code>lambda.start</code>	(Optional) Starting value for the proportion of over-expressed genes among differentially expressed genes. The default value is 0.9.
<code>tolerance</code>	(Optional) The function <code>equi.gene.norm()</code> iteratively estimates an array effect (<code>alpha.hat[i]</code>) for each sample. Algorithm convergence is obtained when the sum of the absolute differences of the array effects from one iteration to the next is less than the specified tolerance value. That is, iterations continue until $\text{sum}(\text{abs}(\text{alpha.hat} - \text{alpha.old})) < \text{tolerance}$ (or, alternatively, when the maximum number of iterations is reached see <code>maxIter</code>). The default value is 1E-3.
<code>maxIter</code>	(Optional)The maximum number of iterations for <code>equi.gene.norm()</code> . The default value is 10.
<code>lemma.outdir</code>	(Optional) The function <code>equi.gene.norm()</code> implements an EM algorithm to estimate array effects and classify genes as non-differentially expressed, under-expressed, or over-expressed. The M-step of this EM algorithm calls the <code>lemma()</code> function in the lemma library to estimate an auxiliary EM algorithm, which actually does the gene classification. The following parameters are passed to <code>lemma()</code> . For further information see the documentation for the lemma package. <code>lemma.outdir</code> specifies the output directory in which results from <code>lemma()</code> estimation are stored. The default value is <code>tempdir()</code> . This will place intermediate files in the R temporary folder.
<code>lemma.tol</code>	(Optional)This is the tolerance parameter for <code>lemma()</code> and it determines how many iterations of the EM algorithm must be performed. The default value is 1E-6.
<code>lemma.maxIts</code>	(Optional)This is the maximum number of iterations for <code>lemma()</code> . The default value is 50000.
<code>lemma.plots</code>	(Optional) This parameter specifies whether the <code>lemma()</code> function should generate diagnostic plots. The default value is FALSE.

Details

The function `equi.gene.norm()` implements a hierarchical mixture model to classify genes as differentially expressed or equivalently expressed and to simultaneously normalize the array data by estimating an array effect for each sample and then subtracting out these array effects. The gene classification step is estimated via the Laplace-approximated expectation-maximization algorithm (LEMMA) for a mixture of Gaussian distributions. This model is implemented in the lemma R library. Included in the output from the `lemma()` function is a prediction for each gene indicating whether it is non-differentially expressed, under-expressed, or over-expressed.

Thus, the EM algorithm implemented in `equi.gene.norm()` consists of iterating between (1) estimating and removing sample-specific array effects, and (2) predicting expression group for each gene via an auxiliary EM algorithm implemented in the `lemma` package. Once the model converges, we estimate a mixed effects model regressing mean expression level (y) of each probe on binary variables indicating over-expression and under-expression (x_o and x_u , respectively).

The function `equi.gene.norm()` returns a list `est.hat` containing the estimates for the model parameters and predicted expression group for each gene. More details are available in the documentation for `est.hat`.

Value

`est.hat` List containing the estimates of `theta.hat` (model parameters) and `e.hat` (prediction of over-expression or under-expression).

Note

This algorithm executes relatively quickly because the implementation in the `lemma` library is very efficient. However, there may be a limit to the number of samples that can be estimated reliably. Larger datasets may cause the algorithm to fail. Our simulations suggests the limit for Windows machines is approximately fifty samples, and the limit for Linux machines is approximately 120 samples. Of course this depends on the particular data and the memory/computational power available. Additionally, several warning messages may be returned when working with larger datasets. Two such warnings are: "In `objective(.par, ...)` : value out of range in 'gammafn'" and "In `sqrt(M[2, 2])` : NaNs produced." These are merely warnings (not errors) and do not appear to affect the accuracy of the results.

Author(s)

Qin LX and Satagopan J.

References

Qin LX and Satagopan J. Normalization method for transcriptional studies of heterogeneous samples - simultaneous array normalization and identification of equivalent expression. *Statistical Applications in Genetics and Molecular Biology* 2009, 8: Article 10.

Haim Bar and Elizabeth Schifano. (2010). `lemma`: Laplace approximated EM Microarray Analysis. R package version 1.3-1. <http://CRAN.R-project.org/package=lemma>

Douglas Bates <bates@stat.wisc.edu> and Martin Maechler <maechler@R-project.org> (2010). `lme4`: Linear mixed-effects models using S4 classes. R package version 0.999375-37. <http://CRAN.R-project.org/package=lme4>

Douglas Bates <bates@stat.wisc.edu> and Martin Maechler <maechler@stat.math.ethz.ch> (2010). `Matrix`: Sparse and Dense Matrix Classes and Methods. R package version 0.999375-46. <http://CRAN.R-project.org/package=Matrix>

Sarkar, Deepayan (2008) `Lattice`: Multivariate Data Visualization with R. Springer, New York.

ISBN 978-0-387-75968-5

Examples

```
#####
# Load data.simulated data      #
#####

data(data.simulated)

#####
# Run equi.gene.norm function.    #
# data.x and data.y are the only required parameters; #
# all others are optional.        #
#####

est.hat.lemma <- equi.gene.norm( data.simulated$data.y, data.simulated$data.x,
                                pi.start = 0.1, lambda.start = 0.9, tolerance = 1E-3, maxIter = 10,
                                lemma.outdir = tempdir(), lemma.tol = 1E-6, lemma.maxIts = 50000, lemma.plots = FALSE )

#####
# Print results from estimation and compare with true values.      #
# The true values are known because the example data are data.simulated. #
#####

# alpha.hat is an n-element list containing the estimated array effect for each of the arrays.
# pi.hat is the estimated proportion of genes that are differentially expressed.
# lambda.hat is the estimated proportion of differentially expressed genes that are over-expressed.
# mu0.hat is the treatment effect of gene expression among over-expressed genes.
# muU.hat is the treatment effect of gene expression among under-expressed genes.
# tau2.hat is the variance of gene effects.
# psi2.hat is the variance of treatment effect for over-expressed genes.
# xi2.hat is the variance of the treatment effect for under-expressed genes.
# sigma2.hat is the variance of measurement error.

est.hat.lemma$theta.hat

# Percent error for parameter estimates
100*(est.hat.lemma$theta.hat$pi.hat - data.simulated$true.values$pi)/data.simulated$true.values$pi
100*(est.hat.lemma$theta.hat$lambda.hat - data.simulated$true.values$lambda)/data.simulated$true.values$lambda
100*(est.hat.lemma$theta.hat$muU.hat - data.simulated$true.values$muU)/data.simulated$true.values$muU
100*(est.hat.lemma$theta.hat$mu0.hat - data.simulated$true.values$mu0)/data.simulated$true.values$mu0
100*(est.hat.lemma$theta.hat$tau2.hat - data.simulated$true.values$tau2)/data.simulated$true.values$tau2
100*(est.hat.lemma$theta.hat$psi2.hat - data.simulated$true.values$psi2)/data.simulated$true.values$psi2
100*(est.hat.lemma$theta.hat$xi2.hat - data.simulated$true.values$xi2)/data.simulated$true.values$xi2
100*(est.hat.lemma$theta.hat$sigma2.hat - data.simulated$true.values$sigma2)/data.simulated$true.values$sigma2

# Percent error for estimated array effects
alpha.compare <- cbind(data.simulated$data.x, data.simulated$true.values$alpha, est.hat.lemma$theta.hat$alpha.hat,
                      100*(est.hat.lemma$theta.hat$alpha.hat - data.simulated$true.values$alpha)/data.simulated$true.values$alpha)
colnames(alpha.compare) <- c("data.simulated$data.x", "True alpha", "alpha.hat", "Percent Error" )
alpha.compare
```

```

# Plot of true array effects versus estimated array effects

Control.count <- length( which( data.simulated$data.x == 0 ) )
Case.count <- length( which( data.simulated$data.x == 1 ) )

plot( x = data.simulated$true.values$alpha, y = est.hat.lemma$theta.hat$alpha.hat,
      xlab = "True alpha", ylab = "Estimated alpha",
      main = "Plot of True alpha vs. Estimated alpha",
      col = c(rep("black",Control.count), rep("blue",Case.count) ) )
abline( a = 0, b = 1, col = "red" )
legend( "bottomright", pch = c(1,1), col = c("black", "blue" ),
       legend = c("Control","Case") )

# Cross-tabulations for predictions of under-expression and over-expression

table( data.simulated$true.values$data.u, est.hat.lemma$e.hat$u.hat )
prop.table(table( data.simulated$true.values$data.u, est.hat.lemma$e.hat$u.hat ))

table( data.simulated$true.values$data.o, est.hat.lemma$e.hat$o.hat )
prop.table(table( data.simulated$true.values$data.o, est.hat.lemma$e.hat$o.hat ))

```

est.hat

List containing the estimates from the hierarchical mixture model.

Description

List containing the estimates from the hierarchical mixture model.

Format

The format is:

List of 2

```

\ $ theta.hat : List of 10
\ $ alpha.hat : num [1:n] 8.14 8.04 7.85 7.19 8.21 ...
\ $ pi.hat : num 0.32
\ $ lambda.hat: num 0.37
\ $ muO.hat : num 0.355
\ $ muU.hat : num -0.0523
\ $ tau2.hat : num 1.25
\ $ psi2.hat : num 4.41e-06
\ $ xi2.hat : num 0.767
\ $ sigma2.hat: num 0.0516
\ $ llh : num -50373

```

$\backslash \$ e.hat$: List of 2
 $\backslash \$ o.hat$: int [1:S] 0 0 0 0 0 0 0 0 1 ...
 $\backslash \$ u.hat$: int [1:S] 1 0 1 0 0 0 0 0 1 0 ...

Details

theta.hat is a 10-element list containing the following:
 $\alpha.hat$ is an n -element list containing the estimated array effect for each of the arrays.
 $\pi.hat$ is the estimated proportion of genes that are differentially expressed.
 $\lambda.hat$ is the estimated proportion of differentially expressed genes that are over-expressed.
 $\mu O.hat$ is the treatment effect of gene expression among over-expressed genes.
 $\mu U.hat$ is the treatment effect of gene expression among under-expressed genes.
 $\tau^2.hat$ is the variance of gene effects.
 $\psi^2.hat$ is the variance of treatment effect for over-expressed genes.
 $\xi^2.hat$ is the variance of the treatment effect for under-expressed genes.
 $\sigma^2.hat$ is the variance of measurement error.
 llh is the log-likelihood.

e.hat is a 2-element list containing the following:
 $o.hat$ is an S -element list with value equal 1 if that probeset is predicted to be *over*-expressed and 0 otherwise.
 $u.hat$ is an S -element list with value equal 1 if that probeset is predicted to be *under*-expressed and 0 otherwise.

Where S is the number of probesets and n is the number of samples in the dataset.

lambda.start	<i>Starting value for proportion of over-expressed genes.</i>
--------------	---

Description

(Optional) Starting value for the proportion of over-expressed genes among differentially expressed genes. The default value is 0.9.

lemma.maxIts	<i>Maximum number of iterations for lemma.</i>
--------------	--

Description

(Optional) This is the maximum number of iterations for lemma(). The default value is 50000.

lemma.outdir	<i>Directory for lemma output.</i>
--------------	------------------------------------

Description

(Optional) The function `equi.gene.norm()` implements an EM algorithm to estimate array effects and classify genes as non-differentially expressed, under-expressed, or over-expressed. The M-step of this EM algorithm calls the `lemma()` function in the lemma library to estimate an auxiliary EM algorithm, which actually does the gene classification. The following parameters are passed to `lemma()`. For further information see the documentation for the lemma package. `lemma.outdir` specifies the output directory in which results from `lemma()` estimation are stored. The default value is `tempdir()`. This will place intermediate files in the R temporary folder.

lemma.plots	<i>A logical value indicating whether lemma should generate plots.</i>
-------------	--

Description

(Optional) This parameter specifies whether the `lemma()` function should generate diagnostic plots. The default value is `FALSE`.

lemma.tol	<i>Tolerance for lemma convergence.</i>
-----------	---

Description

(Optional) This is the tolerance parameter for `lemma()` and it determines how many iterations of the EM algorithm must be performed. The default value is `1E-6`.

maxIter	<i>Maximum number of iterations for EquiNorm.</i>
---------	---

Description

(Optional) The maximum number of iterations for `equi.gene.norm()`. The default value is `10`.

pi.start	<i>Starting value for proportion of differentially expressed genes.</i>
----------	---

Description

(Optional) Starting value for the proportion of differentially expressed genes. The default value is 0.1.

tolerance	<i>Tolerance for EquiNorm convergence.</i>
-----------	--

Description

(Optional) The function `equi.gene.norm()` iteratively estimates an array effect ($\alpha.\hat{[i]}$) for each sample. Algorithm convergence is obtained when the sum of the absolute differences of the array effects from one iteration to the next is less than the specified tolerance value. That is, iterations continue until $\text{sum}(\text{abs}(\alpha.\hat{[i]} - \alpha.\text{old})) < \text{tolerance}$ (or, alternatively, when the maximum number of iterations is reached see `maxIter`). The default value is 1E-3.

Index

*Topic **argument**

- lambda.start, 11
- lemma.maxIts, 11
- lemma.outdir, 12
- lemma.plots, 12
- lemma.tol, 12
- maxIter, 12
- pi.start, 13
- tolerance, 13

*Topic **datasets**

- data.simulated, 4
- data.x, 6
- data.y, 6
- est.hat, 10

- data.simulated, 4
- data.x, 6
- data.y, 6

- equi.gene.norm, 6
- EquiNorm (EquiNorm-package), 2
- EquiNorm-package, 2
- est.hat, 10

- lambda.start, 11
- lemma.maxIts, 11
- lemma.outdir, 12
- lemma.plots, 12
- lemma.tol, 12

- maxIter, 12

- pi.start, 13

- tolerance, 13