Package ‘gaga’

February 1, 2026

Version 2.57.0

Date 2025-02-26

Title GaGa hierarchical model for high-throughput data analysis

Author David Rossell <rosselldavid@gmail.com>.

Maintainer David Rossell <rosselldavid@gmail.com>

Depends R (>= 2.8.0), Biobase, coda, EBarrays, mgcv

Enhances parallel

Description Implements the GaGa model for high-throughput data analysis, including differential expression analysis, supervised gene clustering and classification. Additionally, it performs sequential sample size calculations using the GaGa and LNNGV models (the latter from EBarrays package).

License GPL (>= 2)

biocViews ImmunoOncology, OneChannel, MassSpectrometry, MultipleComparison, DifferentialExpression, Classification

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

git_branch devel

git_last_commit f057de3

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-02-01

Contents

buildPatterns 2
checkfit 3
classpred 4
dcgamma 6
findgenes 7
fitGG 9
forwsimDiffExpr 12

geneclus 14
getpar 16
parest 17
plotForwSim 18
posmeansGG 19
powclasspred 20
powfindgenes 22
print.gagaclus 24
print.gagafit 24
print.gagahyp 25
seqBoundariesGrid 26
simGG 28
simnewsamples 29

Index **31**

`buildPatterns` *Build a matrix with all possible patterns given a number of groups where samples may belong to.*

Description

Creates a matrix indicating which groups are put together under each pattern. The number of possible patterns increases very fast as the number of groups increases. This function provides an easy way to compute all possible patterns. The output of this function is usually used for the `patterns` parameter of the `lmFit` function.

Usage

```
buildPatterns(groups)
```

Arguments

`groups`	Character containing the names of the groups at which samples may belong to. If the output of the function is going to be used in `fitGG` it must match the group levels specified in the `groups` parameter that will be given to `fitGG`.

Examples

```
buildPatterns(groups=c('GroupControl','GroupA','GroupB'))
```

checkfit *Check goodness-of-fit of GaGa and MiGaGa models*

Description

Produces plots to check fit of GaGa and MiGaGa model. Compares observed data with posterior predictive distribution of the model. Can also compare posterior distribution of parameters with method of moments estimates.

Usage

```
checkfit(gg.fit, x, groups, type='data', logexpr=FALSE, xlab, ylab, main, lty, lwd, ...)
```

Arguments

gg.fit	GaGa or MiGaGa fit (object of type gagafit, as returned by fitGG).
x	ExpressionSet, exprSet, data frame or matrix containing the gene expression measurements used to fit the model.
groups	If x is of type ExpressionSet or exprSet, groups should be the name of the column in pData(x) with the groups that one wishes to compare. If x is a matrix or a data frame, groups should be a vector indicating to which group each column in x corresponds to.
type	data checks marginal density of the data; shape checks shape parameter; mean checks mean parameter; shapemean checks the joint of shape and mean parameters
logexpr	If set to TRUE, the expression values are in log2 scale.
xlab	Passed on to plot
ylab	Passed on to plot
main	Passed on to plot
lty	Ignored.
lwd	Ignored.
...	Other arguments to be passed to plot

Details

The routine generates random draws from the posterior and posterior predictive distributions, fixing the hyper-parameters at their estimated value (posterior mean if model was fit with method=='Bayes' or maximum likelihood estimate is model was fit with method=='EBayes').

Value

Produces a plot.

Note

Posterior and posterior predictive checks can lack sensitivity to detect model misfit, since they are susceptible to over-fitting. An alternative is to perform prior predictive checks by generating parameters and data with `simGG`.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

`simGG` to simulate samples from the prior-predictive distribution, `simnewsamples` to generate parameters and observations from the posterior predictive, which is useful to check goodness-of-fit individually a desired gene.

`classpred` *Predict the class that a new sample belongs to.*

Description

Computes the posterior probability that a new sample belongs to each group and classifies it into the group with highest probability.

Usage

```
classpred(gg.fit, xnew, x, groups, prgroups, ngene=100)
```

Arguments

`gg.fit`	GaGa or MiGaGa fit (object of type `gagafit`, as returned by `fitGG`).
`xnew`	Expression levels of the sample to be classified. Only the subset of the genes indicated by ngene is used.
`x`	`ExpressionSet`, `exprSet`, data frame or matrix containing the gene expression measurements used to fit the model.
`groups`	If `x` is of type `ExpressionSet` or `exprSet`, `groups` should be the name of the column in `pData(x)` with the groups that one wishes to compare. If `x` is a matrix or a data frame, `groups` should be a vector indicating to which group each column in x corresponds to.
`prgroups`	Vector specifying prior probabilities for each group. Defaults to equally probable groups.
`ngene`	Number of genes to use to build the classifier. Genes with smaller probability of being equally expressed are selected first.

Details

The classifier weights each gene according to the posterior probability that it is differentially expressed. Hence, adding genes that are unlikely to be differentially expressed does not affect the performance of the classifier, but it does increase the computational cost. All computations are performed by fixing the hyper-parameters to their estimated value (posterior mean if model was fit with `method=='Bayes'` or maximum likelihood estimate is model was fit with `method=='EBayes'`).

Value

List with the following elements:

`d`	Numeric value indicating the group that the new sample is classified into, i.e. where the maximum in `posgroups` is.
`posgroups`	Vector giving the posterior probability that the `xnew` belongs to each of the groups.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

`fitGG`, `parest`

Examples

```
#Not run. Example from the help manual
#library(gaga)
#set.seed(10)
#n <- 100; m <- c(6,6)
#a0 <- 25.5; nu <- 0.109
#balpha <- 1.183; nualpha <- 1683
#probpat <- c(.95,.05)
#xsim <- simGG(n,m,p.de=probpat[2],a0,nu,balpha,nualpha)
#
#ggfit <- fitGG(xsim$x[,c(-6,-12)],groups,patterns=patterns,nclust=1)
#ggfit <- parest(ggfit,x=xsim$x[,c(-6,-12)],groups,burnin=100,alpha=.05)
#
#pred1 <- classpred(ggfit,xnew=xsim$x[,6],x=xsim$x[,c(-6,-12)],groups)
#pred2 <- classpred(ggfit,xnew=xsim$x[,12],x=xsim$x[,c(-6,-12)],groups)
#pred1
#pred2
```

dcgamma *Approximate gamma shape distribution*

Description

dcgamma approximates density of a gamma shape distribution with a gamma density. rcgamma obtains random draws from the approximation. mcgamma computes approximated mean, variance and normalization constant.

Usage

```
dcgamma(x, a, b, c, d, r, s, newton = TRUE)
rcgamma(n, a, b, c, d, r, s, newton = TRUE)
mcgamma(a, b, c, d, r, s, newton = TRUE)
```

Arguments

x	Vector indicating the values at which to evaluate the density.
n	Number of random draws to obtain.
a, b, c, d, r, s	Parameter values.
newton	Set to TRUE to try to locate the mode by taking a few Newton-Raphson steps.

Details

The density of a gamma shape distribution is given by C(a,b,c,d,r,s) (gamma(a*x+d)/gamma(x)^a) (x/(r+s*x))^{a*x+d} x^{b-d-1} exp(-x*c) for x>=0, and 0 otherwise, where C() is the normalization constant. The gamma approximation is Ga(a/2+b-1/2,c+a*log(s/a)). The approximate normalization constant is obtained by taking the ratio of the exact density and the approximation at the maximum, as described in Rossell (2007).

Value

dcgamma returns a vector with approximate density. rcgamma returns a vector with draws from the approximating gamma. mcgamma returns a list with components:

m	Approximate mean
v	Approximate variance
normk	Approximate normalization constant

Note

For general values of the parameters the gamma approximation may be poor. In such a case one could use this function to obtain draws from the proposal distribution in a Metropolis-Hastings step.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

dgamma, rgamma

---

findgenes *Find differentially expressed genes after GaGa or Normal-Normal fit.*

Description

Obtains a list of differentially expressed genes using the posterior probabilities from a GaGa, MiGaGa or Normal-Normal fit. For parametric==TRUE the procedure controls the Bayesian FDR below fdrmax. For parametric==FALSE it controls the estimated frequentist FDR (only available for GaGa).

Usage

```
findgenes(fit, x, groups, fdrmax=.05, parametric=TRUE, B=500)
```

Arguments

fit	Either GaGa/MiGaGa fit (object of class gagafit, as returned by fitGG) or Normal-Normal fit (class nnfit, as returned by fitNN).
x	ExpressionSet, exprSet, data frame or matrix containing the gene expression measurements used to fit the model.
groups	If x is of type ExpressionSet or exprSet, groups should be the name of the column in pData(x) with the groups that one wishes to compare. If x is a matrix or a data frame, groups should be a vector indicating to which group each column in x corresponds to.
fdrmax	Upper bound on FDR.
.	
parametric	Set to TRUE to control the posterior expected FDR below fdrmax. Set to FALSE to estimate the frequentist FDR non-parametrically (only available when fit is of class gagafit).
B	Number of boostrap samples to estimate FDR non-parametrically (ignored if parametric==TRUE)

Details

The Bayes rule to minimize posterior expected FNR subject to posterior expected FDR <=fdrmax declares differentially expressed all genes with posterior probability of being equally expressed below a certain threshold. The value of the threshold is computed exactly for parametric==TRUE, FDR being defined in a Bayesian sense. For parametric==FALSE the FDR is defined in a frequentist sense.

Value

List with components:

truePos	Expected number of true positives.
d	Vector indicating the pattern that each gene is assigned to.
fdr	Frequentist estimated FDR that is closest to fdrmax.
fdrpar	Bayesian FDR. If parametric==TRUE, this is equal to fdrmax. If parametric==FALSE, it's the Bayesian FDR needed to achieve frequentist estimated FDR=fdrmax.
fdrest	Data frame with estimated frequentist FDR for each target Bayesian FDR
fnr	Bayesian FNR
power	Bayesian power as estimated by expected number of true positives divided by the expected number of differentially expressed genes
threshold	Optimal threshold for posterior probability of equal expression (genes with probability < threshold are declared DE)

Author(s)

David Rossell

References

Rossell D. (2009) GaGa: a Parsimonious and Flexible Model for Differential Expression Analysis. Annals of Applied Statistics, 3, 1035-1051.

Yuan, M. and Kendziorski, C. (2006). A unified approach for simultaneous gene clustering and differential expression identification. Biometrics 62(4): 1089-1098.

Muller P, Parmigiani G, Robert C, Rousseau J. (2004) Journal of the American Statistical Association, 99(468): 990-1001.

See Also

fitGG, fitNN, parest

Examples

```
#Not run. Example from the help manual
#library(gaga)
#set.seed(10)
#n <- 100; m <- c(6,6)
#a0 <- 25.5; nu <- 0.109
#balpha <- 1.183; nualpha <- 1683
#probpat <- c(.95,.05)
#xsim <- simGG(n,m,p.de=probpat[2],a0,nu,balpha,nualpha)
#
#ggfit <- fitGG(xsim$x[,c(-6,-12)],groups,patterns=patterns,nclust=1)
#ggfit <- parest(ggfit,x=xsim$x[,c(-6,-12)],groups,burnin=100,alpha=.05)
#
#d <- findgenes(ggfit,xsim$x[,c(-6,-12)],groups,fdrmax=.05,parametric=TRUE)
#dtrue <- (xsim$l[,1]!=xsim$l[,2])
#table(d$d,dtrue)
```

fitGG *Fit GaGa hierarchical model*

Description

fitGG fits GaGa/MiGaGa hierarchical models, either via a fully Bayesian approach or via maximum likelihood.

fitNN fits a normal-normal hierarchical model (wrapper to call emfit in package EBarrays with the LNNMV model). fitNNSingleHyp is the same as fitNN but only considers the pattern that all groups are different from each other.

adjustfitNN corrects a small sample-size bias in the fitNN estimation procedure.

Usage

```
fitGG(x, groups, patterns, equalcv = TRUE, nclust = 1, method =
 "quickEM", B, priorpar, parini, trace = TRUE)

fitNN(x, groups, patterns, B=20, trace=TRUE)

fitNNSingleHyp(x, groups, B=10, trace=TRUE)

adjustfitNN(fit, pitrue, B=5, nsim=3, mc.cores=1)
```

Arguments

x	ExpressionSet, exprSet, data frame or matrix containing the gene expression measurements used to fit the model. For fitNN, x should be in log-scale (in contrast to the input to emfit). fitGG accepts raw and log-scale (as long as all log-measurements remain positive). By default we recommend using log-scale to reduce the influence of outliers.
groups	If x is of type ExpressionSet or exprSet, groups should be the name of the column in pData(x) with the groups that one wishes to compare. If x is a matrix or a data frame, groups should be a vector indicating to which group each column in x corresponds to.
patterns	Matrix indicating which groups are put together under each pattern, i.e. the hypotheses to consider for each gene. colnames(patterns) must match the group levels specified in groups. Defaults to two hypotheses: null hypothesis of all groups being equal and full alternative of all groups being different. The function buildPatterns can be used to construct a matrix with all possible patterns.
equalcv	equalcv==TRUE fits model assuming constant CV across groups. equalcv==FALSE compares cv as well as mean expression levels between groups
nclust	Number of clusters in the MiGaGa model. nclust corresponds to the GaGa model.

method	method=='MH' fits a fully Bayesian model via Metropolis-Hastings posterior sampling. method=='Gibbs' does the same using Gibbs sampling. method=='SA' uses Simulated Annealing to find the posterior mode. method=='EM' finds maximum-likelihood estimates via the expectation-maximization algorithm, but this is currently only implemented for nclust>1. method=='quickEM' is a quicker implementation that only performs 2 optimization steps (see details).
B	Number of iterations to fit the model. For method=='MH' and method=='Gibbs', B is the number of MCMC iterations (defaults to 1000). For method=='SA', B is the number of iterations in the Simulated Annealing scheme (defaults to 200). For method=='EM', B is the maximum number of iterations (defaults to 20).
priorpar	List with prior parameter values. It must have components a.alpha0,b.alpha0,a.nu,b.nu,a.balpha,b and p.probpat. If missing they are set to non-informative values that are usually reasonable for RMA and GCRMA normalized data.
parini	list with components a0, nu, balpha, nualpha, probclus and probpat indicating the starting values for the hyper-parameters. If not specified, a method of moments estimate is used.
trace	For trace==TRUE the progress of the model fitting routine is printed.
fit	nnfit object, as returned by fitNN
pitrue	Grid of true pi values for which to evaluate the MoM estimation bias. See details.
nsim	Number of datasets to simulate for each pitrue value
mc.cores	If package multicore is available, mc.cores specifies the number of cores to be used for parallel computing.

Details

For GaGa/MiGaGa models, an approximation is used to sample faster from the posterior distribution of the gamma shape parameters and to compute the normalization constants (needed to evaluate the likelihood). These approximations are implemented in rcgamma and mcgamma.

The cooling scheme in method=='SA' uses a temperature equal to 1/log(1+i), where i is the iteration number.

The EM implementation in method=='quickEM' is a quick EM algorithm that usually delivers hyper-parameter estimates very similar to those obtained via the slower method=='EM'. Additionally, the GaGa model inference has been seen to be robust to moderate changes in the hyper-parameter estimates in most datasets.

fitNN is a wrapper to emfit in package EBarrays with the LNNMV model. This procedure estimates hyper-parameters using the method of moments (MoM), which typically results in overestimating the proportion of differentially expressed genes, which we denote by pi1. adjustfitNN corrects this bias by repeatedly simulating from the prior predictive of a normal-normal model. Simulations are performed for a grid of pi1 values, so that the expected bias can be evaluated for each of them. The bias is then modeled as a smooth function of pi1 using function gam from package mgcv. Finally, the value of pi1 is bias-adjusted and the posterior probabilities are recomputed using the updated pi1 value.

Value

fitGG returns an object of class gagafit, with components

parest	Hyper-parameter estimates. Only returned if method=='EBayes', for method=='Bayes' one must call the function parest after fitGG
mcmc	Object of class mcmc with posterior draws for hyper-parameters. Only returned if method=='Bayes'.
lhood	For method=='Bayes' it is the log-likelihood evaluated at each MCMC iteration. For method=='EBayes' it is the log-likelihood evaluated at the maximum.
nclust	Same as input argument.
patterns	Same as input argument, converted to object of class gagahyp.

fitNN returns an analogous object of class nnfit. The component nn.fit is the object returned by emfit.

Author(s)

David Rossell

References

Rossell D. (2009) GaGa: a Parsimonious and Flexible Model for Differential Expression Analysis. Annals of Applied Statistics, 3, 1035-1051.

Yuan, M. and Kendziorski, C. (2006). A unified approach for simultaneous gene clustering and differential expression identification. Biometrics 62(4): 1089-1098.

See Also

parest to estimate hyper-parameters and compute posterior probabilities after a GaGa or MiGaGa fit. findgenes to find differentially expressed genes. classpred to predict the group that a new sample belongs to.

Examples

```
library(gaga)
set.seed(10)
n <- 100; m <- c(6,6)
a0 <- 25.5; nu <- 0.109
balpha <- 1.183; nualpha <- 1683
probpat <- c(.95,.05)
xsim <- simGG(n,m,p.de=probpat[2],a0,nu,balpha,nualpha,equalcv=TRUE)
x <- exprs(xsim)

#Frequentist fit: EM algorithm to obtain MLE
groups <- pData(xsim)$group[c(-6,-12)]
patterns <- matrix(c(0,0,0,1),2,2)
colnames(patterns) <- c('group 1','group 2')
gg1 <- fitGG(x[,c(-6,-12)],groups,patterns=patterns,method='EM',trace=FALSE)
gg1 <- parest(gg1,x=x[,c(-6,-12)],groups)
```

gg1

---

forwsimDiffExpr *Forward simulation for differential expression.*

Description

Forward simulation allows to evaluate the expected utility for sequential designs. Here the utility is the expected number of true discoveries minus a sampling cost. The routine simulates future data either from the prior predictive or using a set of pilot data and a GaGa or normal-normal model fit. At each future time point, it computes a summary statistic that will be used to determine when to stop the experiment.

Usage

```
forwsimDiffExpr(fit, x, groups, ngenes, maxBatch, batchSize, fdrmax = 0.05, genelimit, v0thre = 1, B = 1
Bsummary = 100, trace = TRUE, randomSeed, mc.cores=1)
```

Arguments

fit	Either GaGa or MiGaGa fit (object of type gagafit, as returned by fitGG) or Normal-Normal fit (type nnfit returned by fitNN).
x	ExpressionSet, exprSet, data frame or matrix containing the gene expression measurements used to fit the model.
groups	If x is of type ExpressionSet or exprSet, groups should be the name of the column in pData(x) with the groups that one wishes to compare. If x is a matrix or a data frame, groups should be a vector indicating to which group each column in x corresponds to.
ngenes	Number of genes to simulate data for. If x is specified this argument is set to nrow(x) and data is simulated from the posterior predictive conditional on x. If x not specified simulation is from the prior predictive.
maxBatch	Maximum number of batches, i.e. the routine simulates batchSize*maxBatch samples per group.
batchSize	Batch size, i.e. number of observations per group to simulate at each time point. Defaults to ncol(x)/length(unique(groups)).
fdrmax	Upper bound on FDR.
genelimit	Only the genelimit genes with the lowest probability of being equally expressed across all groups will be simulated. Setting this limit can significantly increase the computational speed.
v0thre	Only genes with posterior probability of being equally expressed < v0thre will be simulated. Setting this limit can significantly increase the computational speed.
B	Number of forward simulations.

`Bsummary`	Number of simulations for estimating the summary statistic.
`trace`	For `trace==TRUE` iteration progress is displayed.
`randomSeed`	Integer value used to set random number generator seed. Defaults to `as.numeric(Sys.time())` modulus 10^6.
`mc.cores`	If `multicore` package is available, `mc.cores` indicates the number of cores to use for parallel computing. Currently only used when `fit` is of class `nnfit`.

Details

To improve computational speed hyper-parameters are not re-estimated as new data is simulated.

Value

A `data.frame` with the following columns:

`simid`	Simulation number.
`j`	Time (sample size).
`u`	Expected number of true positives if we were to stop experimentation at this time.
`fdr`	Expected FDR if we were to stop experimentation at this time.
`fnr`	Expected FNR if we were to stop experimentation at this time.
`power`	Expected power (as estimated by E(TP)/E(positives)) if we were to stop experimentation at this time.
`summary`	Summary statistic: increase in expected true positives if we were to obtain one more data batch.

Author(s)

David Rossell.

References

Rossell D., Mueller P. Sequential sample sizes for high-throughput hypothesis testing experiments. http://sites.google.com/site/rosselldavid/home.

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. Annals of Applied Statistics, 2009, 3, 1035-1051.

See Also

`plotForwSim` to plot the simulated trajectories, `fitGG` for fitting a GaGa model, `fitNN` for fitting a normal-normal model, `seqBoundariesGrid` for finding the optimal design based on the forwards simulation output. `powfindgenes` for fixed sample size calculations.

Examples

```
#Simulate data and fit GaGa model
set.seed(1)
x <- simGG(n=20,m=2,p.de=.5,a0=3,nu=.5,balpha=.5,nualpha=25)
gg1 <- fitGG(x,groups=1:2,method='EM')
gg1 <- parest(gg1,x=x,groups=1:2)

#Run forward simulation
fs1 <- forwsimDiffExpr(gg1, x=x, groups=1:2,
maxBatch=2,batchSize=1,fdrmax=0.05, B=100, Bsummary=100, randomSeed=1)

#Expected number of true positives for each sample size
tapply(fs1$u,fs1$time,'mean')

#Expected utility for each sample size
samplingCost <- 0.01
tapply(fs1$u,fs1$time,'mean') - samplingCost*(0:2)

#Optimal sequential design
b0seq <- seq(0,20,length=200); b1seq <- seq(0,40,length=200)
bopt <-seqBoundariesGrid(b0=b0seq,b1=b1seq,forwsim=fs1,samplingCost=samplingCost,powmin=0)
bopt <- bopt$opt

plot(fs1$time,fs1$summary,xlab='Additional batches',ylab='E(newly discovered DE genes)')
abline(bopt['b0'],bopt['b1'])
text(.2,bopt['b0'],'Continue',pos=3)
text(.2,bopt['b0'],'Stop',pos=1)
```

`geneclus` *Cluster genes into expression patterns.*

Description

Performs supervised gene clustering. Clusters genes into the expression pattern with highest posterior probability, according to a GaGa or MiGaGa fit.

Usage

```
geneclus(gg.fit, method='posprob')
```

Arguments

`gg.fit`	GaGa or MiGaGa fit (object of type `gagafit`, as returned by `fitGG`).
`method`	For `method=='posprob'` samples are assigned to pattern with highest posterior probability, and for `method=='likelihood'` to the pattern with highest likelihood (e.g. assuming equal a priori prob for all patterns)

Details

Each gene is assigned to the pattern with highest posterior probability. This is similar to routine `findgenes`, which also assigns genes to the pattern with highest posterior probability, although `findgenes` applies an FDR-based correction i.e. tends to assign more genes to the null pattern of no differential expression.

Value

List with components:

`d`	Vector indicating the pattern that each gene is assigned to.
`posprob`	Vector with posterior probabilities of the assigned patterns.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

`fitGG`, `parest`

Examples

```
#Not run. Example from the help manual
#library(gaga)
#set.seed(10)
#n <- 100; m <- c(6,6)
#a0 <- 25.5; nu <- 0.109
#balpha <- 1.183; nualpha <- 1683
#probpat <- c(.95,.05)
#xsim <- simGG(n,m,p.de=probpat[2],a0,nu,balpha,nualpha)
#
#ggfit <- fitGG(xsim$x[,c(-6,-12)],groups,patterns=patterns,nclust=1)
#ggfit <- parest(ggfit,x=xsim$x[,c(-6,-12)],groups,burnin=100,alpha=.05)
#
#dclus <- geneclus(ggfit)  #not use FDR correction
#dfdr <- findgenes(ggfit,xsim$x[,c(-6,-12)],groups,fdrmax=.05,parametric=TRUE) #use FDR correction
#table(dfdr$d,dclus$d) #compare results
```

getpar *Extract hyper-parameter estimates from a gagafit or nnfit object*

Description

Extracts the hyper-parameter estimates from a `gagafit` or `nnfit` object and puts them in a list.

Usage

```
getpar(fit)
```

Arguments

`fit`	Object of class `gagafit` or `nnfit`.

Details

This routine simply evaluates the component `parest` from a `gagafit` or `nnfit` object, which causes an error if this component is not available. This routine is used internally by a number of other routines.

Value

For `gagafit` objects, a list with components:

`a0`	Estimated value of hyper-parameter `a0`
`nu`	Estimated value of hyper-parameter `nu`
`balpha`	Estimated value of hyper-parameter `balpha`
`nualpha`	Estimated value of hyper-parameter `nualpha`
`probclus`	Estimated cluster probabilities
`probpat`	Estimated prior probability of each expression pattern

For `nnfit` objects, a vector with elements `mu0`, `tau02`, `v0`, `sigma02`, `probclus` and `probpat`. These are the hierarchical N(mu0,tau0^2) * IG(.5*v0,.5*v0*sigma0^2) prior parameter estimates.

Author(s)

David Rossell

References

Rossell D. (2009) GaGa: a Parsimonious and Flexible Model for Differential Expression Analysis. Annals of Applied Statistics, 3, 1035-1051.

Yuan, M. and Kendziorski, C. (2006). A unified approach for simultaneous gene clustering and differential expression identification. Biometrics 62(4): 1089-1098.

See Also

fitGG, fitNN, parest

---

parest — *Parameter estimates and posterior probabilities of differential expression for GaGa and MiGaGa model*

Description

Obtains parameter estimates and posterior probabilities of differential expression after a GaGa or MiGaGa model has been fit with the function fitGG.

Usage

```
parest(gg.fit, x, groups, burnin, alpha=.05)
```

Arguments

gg.fit	GaGa or MiGaGa fit (object of type gagafit, as returned by fitGG).
x	ExpressionSet, exprSet, data frame or matrix containing the gene expression measurements used to fit the model.
groups	If x is of type ExpressionSet or exprSet, groups should be the name of the column in pData(x) with the groups that one wishes to compare. If x is a matrix or a data frame, groups should be a vector indicating to which group each column in x corresponds to.
burnin	Number of MCMC samples to discard. Ignored if gg.fit was fit with the option method=='EBayes'.
alpha	If gg.fit was fit with the option method=='Bayes', parest also computes 1-alpha posterior credibility intervals.

Details

If gg.fit was fit via MCMC posterior sampling (option method=='Bayes'), parest discards the first burnin iterations and uses the rest to obtain point estimates and credibility intervals for the hyper-parameters. To compute posterior probabilities of differential expression the hyper-parameters are fixed to their estimated value, i.e. not averaged over MCMC iterations.

Value

An object of class gagafit, with components:

parest	Hyper-parameter estimates.
mcmc	Object of class mcmc with posterior draws for hyper-parameters. Only returned if method=='Bayes'.
lhood	For method=='Bayes' it is the posterior mean of the log-likelihood. For method=='EBayes' it is the log-likelihood evaluated at the maximum.

nclust	Number of clusters.
patterns	Object of class gagahyp indicating which hypotheses (expression patterns) were tested.
pp	Matrix with posterior probabilities of differential expression for each gene. Genes are in rows and expression patterns are in columns (e.g. for 2 hypotheses, 1st column is the probability of the null hypothesis and 2nd column for the alternative).

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

fitGG to fit a GaGa or MiGaGa model, findgenes to find differentially expressed genes and posmeansGG to obtain posterior expected expression values. classpred performs class prediction.

Examples

```
#Not run
#library(EBarrays); data(gould)
#x <- log(exprs(gould)[,-1])  #exclude 1st array
#groups <- pData(gould)[-1,1]
#patterns <- rbind(rep(0,3),c(0,0,1),c(0,1,1),0:2) #4 hypothesis
#gg <- fitGG(x,groups,patterns,method='EBayes')
#gg
#gg <- parest(gg,x,groups)
#gg
```

---

plotForwSim *Plot forward simulation trajectories*

Description

Produces plot to visualize simulated trajectories of the summary statistic as a function of the number of additional data batches, i.e. the output produced by forwsimDiffExpr.

Usage

```
plotForwSim(fs,xlab="Number of additional batches per group",ylab="Expected increase in True Positives
```

Arguments

`fs`	Forward simulation results, as output by `forwsimDiffExpr`.
`xlab`	x-axis label
`ylab`	y-axis label
`col`	line color
`lty`	line type
`...`	Other arguments to be passed to `plot`

Value

Produces a plot.

Author(s)

David Rossell

References

Rossell D., Mueller P. Sequential sample sizes for high-throughput hypothesis testing experiments. http://sites.google.com/site/rosselldavid/home.

See Also

`forwsimDiffExpr`

---

posmeansGG *Gene-specific posterior means*

Description

Computes posterior means for the gene expression levels using a GaGa or MiGaGa model.

Usage

```
posmeansGG(gg.fit, x, groups, sel, underpattern)
```

Arguments

`gg.fit`	GaGa or MiGaGa fit (object of type `gagafit`, as returned by `fitGG`).
`x`	`ExpressionSet`, `exprSet`, data frame or matrix containing the gene expression measurements used to fit the model.
`groups`	If `x` is of type `ExpressionSet` or `exprSet`, `groups` should be the name of the column in `pData(x)` with the groups that one wishes to compare. If `x` is a matrix or a data frame, `groups` should be a vector indicating to which group each column in x corresponds to.

`sel` Numeric vector with the indexes of the genes we want to draw new samples for (defaults to all genes). If a logical vector is indicated, it is converted to `(1:nrow(x))[sel]`.

`underpattern` Expression pattern assumed to be true (defaults to last pattern in `gg.fit$patterns`). Posterior means are computed under this pattern. For example, if only the null pattern that all groups are equal and the full alternative that all groups are different are considered, `underpattern=1` returns the posterior means under the assumption that groups are different from each other (`underpattern=0` returns the same mean for all groups).

Details

The posterior distribution of the mean parameters actually depends on the gene-specific shape parameter(s), which is unknown. To speed up computations, a gamma approximation to the shape parameter posterior is used (see `rcgamma` for details) and the shape parameter is fixed to its mode a posteriori.

Value

Matrix with mean expression values a posteriori, for each selected gene and each group. Genes are in rows and groups in columns.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

`fitGG` for fitting GaGa and MiGaGa models, `parest` for computing posterior probabilities of each expression pattern.

`powclasspred` *Expected probability that a future sample is correctly classified.*

Description

Estimates posterior expected probability that a future sample is correctly classified when performing class prediction. The estimate is obtained via Monte Carlo simulation from the posterior predictive.

Usage

```
powclasspred(gg.fit, x, groups, prgroups, v0thre=1, ngene=100, B=100)
```

Arguments

gg.fit	GaGa or MiGaGa fit (object of type gagafit, as returned by fitGG).
x	ExpressionSet, exprSet, data frame or matrix containing the gene expression measurements used to fit the model.
groups	If x is of type ExpressionSet or exprSet, groups should be the name of the column in pData(x) with the groups that one wishes to compare. If x is a matrix or a data frame, groups should be a vector indicating to which group each column in x corresponds to.
prgroups	Vector specifying prior probabilities for each group. Defaults to equally probable groups.
v0thre	Only genes with posterior probability of being equally expressed below v0thre are used.
ngene	Number of genes to use to build the classifier. Genes with smaller probability of being equally expressed are selected first.
B	Number of Monte Carlo samples to be used.

Details

The routine simulates future samples (microarrays) from the posterior predictive distribution of a given group (e.g. control/cancer). Then it computes the posterior probability that the new sample belongs to each of the groups and classifies the sample into the group with highest probability. This process is repeated B times, and the proportion of correctly classified samples is reported for each group. The standard error is obtained via the usual normal approximation (i.e. SD/B). The overall probability of correct classification is also provided (i.e. for all groups together), but using a more efficient variant of the algorithm. Instead of reporting the observed proportion of correctly classified samples, it reports the expected proportion of correctly classified samples (i.e. the average posterior probability of the class that the sample is assigned to).

Value

List with components:

ccall	Estimated expected probability of correctly classifying a future sample.
seccall	Estimated standard error of ccall.
ccgroup	Vector with the estimated probability of correctly classifying a sample from each group.
segroup	Estimated standard error of ccgroup.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

classpred, fitGG, parest. See powfindgenes for differential expression power calculations.

---

powfindgenes *Power computations for differential expression*

Description

powfindgenes evaluates the posterior expected number of true positives (e.g. true gene discoveries) if one were to obtain an additional batch of data. It uses either a GaGa or a normal-normal model fit on a pilot data set.

Usage

```
powfindgenes(fit, x, groups, batchSize = 1, fdrmax = 0.05, genelimit,
v0thre = 1, B = 1000, mc.cores=1)
```

Arguments

fit	GaGa/MiGaGa or normal-normal model fit using pilot data x. It must be an object either of type gagafit (see fitGG) or nnfit (see fitNN).
x	ExpressionSet, exprSet, data frame or matrix containing the gene expression measurements used to fit the model.
groups	If x is of type ExpressionSet or exprSet, groups should be the name of the column in pData(x) with the groups that one wishes to compare. If x is a matrix or a data frame, groups should be a vector indicating to which group each column in x corresponds to.
batchSize	Number of additional samples to obtain per group.
fdrmax	Upper bound on FDR.
.	
genelimit	Only the genelimit genes with the lowest probability of being equally expressed across all groups will be simulated. Setting this limit can significantly increase the computational speed.
v0thre	Only genes with posterior probability of being equally expressed < v0thre will be simulated. Setting this limit can significantly increase the computational speed.
B	Number of simulations from the GaGa predictive distribution to be used to estimate the posterior expected number of true positives.
mc.cores	If multicore package is available, mc.cores indicates the number of cores to use for parallel computing. Currently only used when fit is of class nnfit.

Details

The routine simulates data from the posterior predictive distribution of a GaGa or normal-normal model. That is, first it simulates parameter values (differential expression status, mean expression levels etc.) from the posterior distribution. Then it simulates data using the parameter values drawn from the posterior. Finally the simulated data is used to determine the differential status of each gene, controlling the Bayesian FDR at the `fdrmax` level, as implemented in `findgenes`. As the differential expression status is known for each gene, one can evaluate the number of true discoveries in the reported gene list.

In order to improve speed, hyper-parameters are not re-estimated when computing posterior probabilities for the posterior predictive simulated data.

Value

`m`	Posterior expected number of true positives (as estimated by the sample mean of `B` simulations)
`s`	Standard error of the estimate i.e. SD of the simulations/sqrt(B)

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

`findgenes`, `fitGG`, `fitNN`, `parest`. See `powclasspred` for power calculations for sample classification.

Examples

```
#Simulate data and fit GaGa model
set.seed(1)
x <- simGG(n=20,m=2,p.de=.5,a0=3,nu=.5,balpha=.5,nualpha=25)
gg1 <- fitGG(x,groups=1:2,method='EM')
gg1 <- parest(gg1,x=x,groups=1:2)

#Expected nb of TP for 1 more sample per group
powfindgenes(gg1,x=x,groups=1:2,batchSize=1,fdrmax=.05)$m

#Expected nb of TP for 10 more samples per group
powfindgenes(gg1,x=x,groups=1:2,batchSize=10,fdrmax=.05)$m
```

print.gagaclus — *Print an object of class gagaclus*

Description

Prints an object of class gagaclus, which contains the result of clustering genes into expression patterns.

Usage

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

Arguments

x	Object of type gagaclus.
...	Other arguments to be passed on to the generic print function.

Value

Displays the expression patterns and the number of genes classified into each of them.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

fitGG, geneclus

print.gagafit — *Print an object of class gagafit or nnfit*

Description

Prints an object of class gagafit (as returned by fitGG or parest) or class nnfit. Provides general information and hyper-parameter estimates, if available.

Usage

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

Arguments

x	Object of type gagafit or nnfit.
...	Other arguments to be passed on to the generic print function.

Details

fitGG does not create a complete gagafit object. The complete object is returned by parest, which computes the posterior probabilities of differential expression and obtain hyper-parameter estimates (these are only provided by fitGG when the option method='EBayes' is used).

Value

Prints number of genes, hypotheses, details about the model fitting and hyper-parameter estimates (when available).

Author(s)

David Rossell

See Also

fitGG, fitNN, parest

---

print.gagahyp *Print an object of class gagahyp*

Description

Prints an object of class gagahyp, which contains information on the hypotheses (expression patterns) from a GaGa or MiGaGa model.

Usage

```
## S3 method for class 'gagahyp'
print(x, probpat=NA, ...)
```

Arguments

x	Object of type gagahyp.
probpat	Vector with either estimated probabilities of each hypothesis, or with number of genes classified into each expression pattern.
...	Other arguments to be passed on to the generic print function.

Value

Prints hypotheses. When available, also displays estimated proportion of genes following each expression pattern or the number of genes classified into each expression pattern.

Author(s)

David Rossell

References

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. http://rosselldavid.googlepages.com.

See Also

fitGG, geneclus

---

seqBoundariesGrid — *Evaluate expected utility for parametric sequential stopping boundaries.*

Description

Estimate the expected utility for sequential boundaries parameterized by (b0,b1). Expected utility is estimated on a grid of (b0,b1) values based on a forward simulation output such as that generated by the function forwsimDiffExpr.

Usage

```
seqBoundariesGrid(b0, b1, forwsim, samplingCost, powmin = 0, f = "linear", ineq = "less")
```

Arguments

b0	Vector with b0 values. Expected utility is evaluated for a grid defined by all combinations of (b0,b1) values.
b1	Vector with b1 values.
forwsim	data.frame with forward simulation output, such as that returned by the function forwsimDiffExpr. It must have columns named simid, time, u, fdr, fnr, power and summary. See forwsimDiffExpr for details on the meaning of each column.

`samplingCost` Cost of obtaining one more data batch, in terms of the number of new truly differentially expressed discoveries that would make it worthwhile to obtain one more data batch.

`powmin` Constraint on power. Optimization chooses the optimal `b0`, `b1` satisfying power>=powermin (if such `b0`,`b1` exists).

`f` Parametric form for the stopping boundary. Currently only 'linear' and 'invsqrt' are implemented. For 'linear', the boundary is `b0+b1*time`. For 'invsqrt', the boundary is `b0+b1/sqrt(time)`, where time is the sample size measured as number of batches.

`ineq` For `ineq=='less'` the trial stops when `summary` is below the stopping boundary. This is appropriate whenever `summary` measures the potential benefit of obtaining one more data batch. For `ineq=='greater'` the trial stops when `summary` is above the stopping boundary. This is approapriate whenever `summary` measures the potential costs of obtaining one more data batch.

Details

Intuitively, the goal is to stop collecting new data when the expected benefit of obtaining one more data batch is small, i.e. below a certain boundary. We consider two simple parametric forms for such a boundary (linear and inverse square root), which allows to easily evaluate the expected utility for each boundary within a grid of parameter values. The optimal boundary is defined by the parameter values achieving the largest expected utility, restricted to parameter values with an estimated power greater or equal than `powmin`. Here power is defined as the expected number of true discoveries divided by the expected number of differentially expressed entities.

The routine evaluates the expected utility, as well as expected FDR, FNR, power and sample size for each specified boundary, and also reports the optimal boundary.

Value

A list with two components:

`opt` Vector with optimal stopping boundary (`b`), estimated expected utility (`u`), false discovery rate (`fdr`), false negative rate (`fnr`), power (`power`) and the expected sample size measured as the number of batches (`time`).

`grid` `data.frame` with all evaluated boundaries (columns `b0` and `b1`) and their respective estimated expected utility, false discovery rate, false negative rate, power and expected sample size (measured as the number of batches).

Author(s)

David Rossell.

References

Rossell D., Mueller P. Sequential sample sizes for high-throughput hypothesis testing experiments. http://sites.google.com/site/rosselldavid/home.

Rossell D. GaGa: a simple and flexible hierarchical model for microarray data analysis. Annals of Applied Statistics, 2009, 3, 1035-1051.

See Also

forwsimDiffExpr

---

simGG *Prior predictive simulation*

Description

simGG simulates parameters and data from the prior-predictive of GaGa/ MiGaGa models with several groups, fixing the hyper-parameters.

simLNN simulates from a log-normal normal with gene-specific variances (LNNMV in package EBarrays). simNN returns the log observations.

Usage

```
simGG(n, m, p.de=.1, a0, nu, balpha, nualpha, equalcv = TRUE, probclus
= 1, a = NA, l = NA, useal = FALSE)

simLNN(n, m, p.de=0.1, mu0, tau0, v0, sigma0)

simNN(n, m, p.de=0.1, mu0, tau0, v0, sigma0)
```

Arguments

n	Number of genes.
m	Vector indicating number of observations to be simulated for each group.
p.de	Probability that a gene is differentially expressed.
a0, nu	Mean expression for each gene is generated from 1/rgamma(a0,a0/nu) if probclus is of length 1, and from a mixture if length(probclus)>1.
balpha, nualpha	Shape parameter for each gene is generated from rgamma(balpha,balpha/nualpha).
equalcv	If equalcv==TRUE the shape parameter is simulated to be constant across groups.
probclus	Vector with the probability of each component in the mixture. Set to 1 for the GaGa model.
a, l	Optionally, if useal==TRUE the parameter values are not generated, only the data is generated. a is a matrix with the shape parameters of each gene and group and l is a matrix with the mean expressions.
useal	For useal==TRUE the parameter values specified in a and l are used, instead of being generated.
mu0, tau0	Gene-specific means arise from N(mu0,tau0^2)
v0, sigma0	Gene-specific variances arise from IG(.5*nu0,.5*nu0*sigma0^2)

Details

For the GaGa model, the shape parameters are actually drawn from a gamma approximation to their posterior distribution. The function rcgamma implements this approximation.

Value

Object of class 'ExpressionSet'. Expression values can be accessed via `exprs(object)` and the parameter values used to generate the expression values can be accessed via `fData(object)`.

Note

Currently, the routine only implements prior predictive simulation for the 2 hypothesis case.

Author(s)

David Rossell

References

Rossell D. (2009) GaGa: a Parsimonious and Flexible Model for Differential Expression Analysis. Annals of Applied Statistics, 3, 1035-1051.

Yuan, M. and Kendziorski, C. (2006). A unified approach for simultaneous gene clustering and differential expression identification. Biometrics 62(4): 1089-1098.

See Also

`simnewsamples` to simulate from the posterior predictive, `checkfit` for graphical posterior predictive checks.

Examples

```
#Not run. Example from the help manual
#library(gaga)
#set.seed(10)
#n <- 100; m <- c(6,6)
#a0 <- 25.5; nu <- 0.109
#balpha <- 1.183; nualpha <- 1683
#probpat <- c(.95,.05)
#xsim <- simGG(n,m,p.de=probpat[2],a0,nu,balpha,nualpha)
#
#plot(density(xsim$x),main='')
#plot(xsim$l,xsim$a,ylab='Shape',xlab='Mean')
```

---

simnewsamples *Posterior predictive simulation*

Description

Posterior and posterior predictive simulation for GaGa/MiGaGa and Normal-Normal models.

Usage

```
simnewsamples(fit, groupsnew, sel, x, groups)
```

Arguments

`fit`	Either GaGa or MiGaGa fit (object of type `gagafit`, as returned by `fitGG`) or Normal-Normal fit (type `nnfit` returned by `fitNN`).
`groupsnew`	Vector indicating the group that each new sample should belong to. `length(groupsnew)` is the number of new samples that will be generated.
`sel`	Numeric vector with the indexes of the genes we want to draw new samples for (defaults to all genes). If a logical vector is indicated, it is converted to `(1:nrow(x))[sel]`. For the Normal-Normal model this argument is ignored.
`x`	`ExpressionSet`, `exprSet`, data frame or matrix containing the gene expression measurements used to fit the model.
`groups`	If `x` is of type `ExpressionSet` or `exprSet`, `groups` should be the name of the column in `pData(x)` with the groups that one wishes to compare. If `x` is a matrix or a data frame, `groups` should be a vector indicating to which group each column in x corresponds to.

Details

For GaGa/MiGaGa models, the shape parameters are actually drawn from a gamma approximation to their posterior distribution. The function `rcgamma` implements this approximation.

In order to be consistent with the LNNGV model implemented in emfit (package EBarrays), for the Normal-Normal model the variance is drawn from an inverse gamma approximation to its marginal posterior (obtained by plugging in the group means, see EBarrays vignette for details).

Value

Object of class 'ExpressionSet'. Expression values can be accessed via `exprs(object)` and the parameter values used to generate the expression values can be accessed via `fData(object)`.

Author(s)

David Rossell

References

Rossell D. (2009) GaGa: a Parsimonious and Flexible Model for Differential Expression Analysis. Annals of Applied Statistics, 3, 1035-1051.

Yuan, M. and Kendziorski, C. (2006). A unified approach for simultaneous gene clustering and differential expression identification. Biometrics 62(4): 1089-1098.

See Also

`checkfit` for posterior predictive plot, `simGG` for prior predictive simulation.

Index

∗ **design**
forwsimDiffExpr, 12
seqBoundariesGrid, 26
∗ **distribution**
checkfit, 3
dcgamma, 6
posmeansGG, 19
simGG, 28
simnewsamples, 29
∗ **htest**
classpred, 4
findgenes, 7
forwsimDiffExpr, 12
geneclus, 14
powclasspred, 20
powfindgenes, 22
seqBoundariesGrid, 26
∗ **logic**
buildPatterns, 2
∗ **models**
checkfit, 3
classpred, 4
findgenes, 7
fitGG, 9
geneclus, 14
getpar, 16
parest, 17
posmeansGG, 19
powclasspred, 20
powfindgenes, 22
simGG, 28
simnewsamples, 29
∗ **plots**
plotForwSim, 18
∗ **print**
print.gagaclus, 24
print.gagafit, 24
print.gagahyp, 25

adjustfitNN *(fitGG)*, 9

buildPatterns, 2

checkfit, 3, *29, 30*
classpred, 4, *11*, *18*, *22*

dcgamma, 6
dgamma, *7*

findgenes, 7, *11*, *18*, *23*
fitGG, *5*, *8*, 9, *13*, *15*, *17, 18*, *20*, *22–26*
fitNN, *8*, *13*, *17*, *23*, *25*
fitNN *(fitGG)*, 9
fitNNSingleHyp *(fitGG)*, 9
forwsimDiffExpr, 12, *19*, *28*

geneclus, 14, *24*, *26*
getpar, 16

mcgamma *(dcgamma)*, 6

parest, *5*, *8*, *11*, *15*, *17*, 17, *20*, *22, 23*, *25*
plotForwSim, *13*, 18
posmeansGG, *18*, 19
powclasspred, 20, *23*
powfindgenes, *13*, *22*, 22
print.gagaclus, 24
print.gagafit, 24
print.gagahyp, 25
print.nnfit *(print.gagafit)*, 24

rcgamma *(dcgamma)*, 6
rgamma, *7*

seqBoundariesGrid, *13*, 26
simGG, *4*, 28, *30*
simLNN *(simGG)*, 28
simnewsamples, *4*, *29*, 29
simNN *(simGG)*, 28