Package ‘pickgene’

March 29, 2017

Version 1.46.0
Author Brian S. Yandell <yandell@stat.wisc.edu>
Title Adaptive Gene Picking for Microarray Expression Data Analysis
Description Functions to Analyze Microarray (Gene Expression) Data.
Maintainer Brian S. Yandell <yandell@stat.wisc.edu>
License GPL (>= 2)
Imports graphics, grDevices, MASS, stats, utils
URL http://www.stat.wisc.edu/~yandell/statgen
biocViews Microarray, DifferentialExpression
NeedsCompilation no

R topics documented:

em.ggb ................................................................. 1
model.pickgene .................................................. 3
oddsplot ......................................................... 4
pickgene .......................................................... 5
robustscale ..................................................... 7
Simulation.pickgene .......................................... 8

Index

em.ggb EM calculation for Gamma-Gamma-Bernoulli Model

Description

The function plots contours for the odds that points on microarray show differential expression
between two conditions (e.g. Cy3 and Cy5 dye channels on the same microarray).

Usage

em.ggb(x, y, theta, start = c(2,1.2,2.7), pprior = 2,
printit = FALSE, tol = 1e-9, offset = 0 )
Arguments

x  
   first condition expression levels

y  
   second condition expression levels

theta  
   four parameters a, a0, nu, p

start  
   starting estimates for theta

pprior  
   Beta hyperparameter for prob p of differential expression

printit  
   print iterations if TRUE

tol  
   parameter tolerance for convergence

offset  
   offset added to xx and yy before taking log (can help with negative adjusted values)

Details

Fit Gamma/Gamma/Bernoulli model (equal marginal distributions) The model has spot intensities x \sim \text{Gamma}(a,b); y \sim \text{Gamma}(a,c). The shape parameters b and c are \sim \text{Gamma}(a0,nu). With probability p, b = c; otherwise b \neq c. All spots are assumed to be independent.

Value

Four parameter vector theta after convergence.

Author(s)

Michael Newton

References


See Also

oddsplot

Examples

```r
## Not run:
em.ggb( x, y )
## End(Not run)
```
model.pickgene

---

**model.pickgene**  
*Create Model Matrix for Orthogonal Contrasts*

**Description**

The function created a model matrix of orthogonal contrasts to be used by pickgene.

**Usage**

```r
model.pickgene(faclevel, facnames = letters[seq(length(faclevel))],
contrasts.fac = "contr.poly", collapse = "+", show =
NULL, renorm = 1, modelexpr = formula(paste("-",
paste(facnames, collapse = collapse))),
contrasts.list = contr.list)
```

**Arguments**

- `faclevel` vector with number of levels for each factor
- `facnames` vector of factor names (default = "a", "b", ...)
- `contrasts.fac` vector of contrast types
- `collapse` "+" for additive model, "*" for full model with interactions
- `show` vector of contrast numbers to show (default is all)
- `renorm` vector to renormalize contrasts (e.g., use `sqrt(2)` to turn two-condition contrast into fold change)
- `modelexpr` model formula
- `contrasts.list` list of contrasts indexed by facnames

**Details**

Creates a model matrix data frame with first column having all 1’s and other columns having contrasts.

**Value**

Result of call to `model.matrix`

**Author(s)**

Brian Yandell

**See Also**

- `model.matrix`

**Examples**

```r
model.pickgene(c(2,3), c("sex","genotype"))
```
oddplot  

**Odds Plot for Differential Microarray Expression**

### Description

The function plots contours for the odds that points on microarray show differential expression between two conditions (e.g. Cy3 and Cy5 dye channels on the same microarray).

### Usage

```r
oddplot(x, y, theta, by.level = 10, rotate = FALSE, offset = 0, main = "", xlab = xlabs, ylab = ylabs, col = NULL, cex = c(0.25, 0.75), shrink = FALSE, lims = range(c(x, y)))
```

### Arguments

- **x**: first condition expression levels
- **y**: second condition expression levels
- **theta**: four parameters from `em.ggb`
- **by.level**: odds plot contours increase by this level
- **rotate**: rotate to average versus ratio if TRUE, otherwise plot conditions against each other
- **offset**: offset for log transform
- **main**: main title for plot
- **xlab**: horizontal axis label (default if `rotate` is FALSE, `Average Intensity` otherwise)
- **ylab**: vertical axis label (default if `rotate` is FALSE, `Cy3 / Cy5` otherwise)
- **col**: color of points (if NULL, use black for non-changing points, blue for changing points)
- **cex**: character expansion (use `rep(.25, 2)` to have all points the same size)
- **shrink**: use shrinkage on expression levels if TRUE (default is FALSE)
- **lims**: limits for plot area

### Details

Fit Gamma/Gamma/Bernoulli model (equal marginal distributions) The model has spot intensities

\[ x \sim \text{Gamma}(a,b); \ y \sim \text{Gamma}(a,c). \]

The shape parameters \( b \) and \( c \) are \( \sim \text{Gamma}(a_0,\nu) \). With probability \( p \), \( b = c \); otherwise \( b \neq c \). All spots are assumed to be independent.

### Value

Log odds for all points in original order.

### Author(s)

Michael Newton
pickgene

References

variability of expression ratios: improving statistical inference about gene expression changes from
microarray data.” J Computational Biology 00: 000-000.

See Also

em.ggb

Examples

## Not run:
oddsplot( x, y )

## End(Not run)

---

pickgene  

Plot and Pick Genes based on Differential Expression

Description

The function picks plots the average intensity versus linear contrasts (currently linear, quadratic up
to cubic) across experimental conditions. Critical line is determine according to Bonferroni-like
multiple comparisons, allowing SD to vary with intensity.

Usage

pickgene(data, geneID = 1:nrow(data), overalllevel = 0.05,
npickgene = -1, marginal = FALSE, rankbased = TRUE,
allrank = FALSE, meanrank = FALSE, offset = 0,
modelmatrix = model.pickgene(faclevel, facnames,
contrasts.fac, collapse, show, renorm), faclevel =
col(data), facnames =
letters[seq(length(faclevel))], contrasts.fac =
"contr.poly", show = NULL, main = "", renorm = 1,
drop.negative = FALSE, plotit = npickgene < 1, mfrow
= c(nr, nc), mfcol = NULL, ylab = paste(shownames,
"Trend"), ...)

Arguments

data  
data matrix

geneID  
gene identifier (default 1:nrow(x))

overalllevel  
overall significance level (default 0.05)

npickgene  
number of genes to pick (default -1 allows automatic selection)

marginal  
additive model if TRUE, include interactions if FALSE

rankbased  
use ranks if TRUE, log tranform if FALSE

allrank  
rank all chips together if true, otherwise rank separately

meanrank  
show mean abundance as rank if TRUE
offset offset for log transform
model.matrix model matrix with first row all 1's and other rows corresponding to design con-
trasts; automatically created by call to model1.pickgene if omitted
fac.level number of factor levels for each factor
fac.names factor names
contrasts.fac type of contrasts
show vector of contrast numbers to show (default is all)
main vector of main titles for plots (default is none)
renorm vector to renormalize contrasts (e.g. use sqrt(2) to turn two-condition contrast into fold change)
drop.negative drop negative values in log transform
plotit plot if TRUE
mfrow par() plot arrangement by rows (default up to 6 per page; set to NULL to not change)
mfcol par() plot arrangement by columns (default is NULL)
ylab vertical axis labels
... parameters for robustscale

Details

Infer genes that differentially express across conditions using a robust data-driven method. Adjusted gene expression levels $A$ are replaced by \( \text{qnorm} \left( \text{rank}(A) \right) \), followed by \text{robustscale} estimation of center and spread. Then Bonferroni-style gene by gene tests are performed and displayed graphically.

Value

Data frame containing significant genes with the following information:

- **pick** data frame with picked genes
- **score** data frame with center and spread for plotting

Each of these is a list with elements for each contrast. The **pick** data frame elements have the following information:

- **probe** gene identifier
- **average** average gene intensity
- **fold1** positive fold change
- **fold2** negative fold change
- **pvalue** Bonferroni-corrected p-value

The **score** data frame elements have the following:

- **x** mean expression level (antilog scale)
- **y** contrast (antilog scale)
- **center** center for contrast
- **scale** scale (spread) for contrast
- **lower** lower test limit
- **upper** upper test limit
robustscale

Author(s)
Yi Lin and Brian Yandell

References

See Also
pickgene

Examples
```r
## Not run:
pickgene( data )
## End(Not run)
```

robustscale

Robust Estimation of Median (center) and MAD (scale)

Description
Smoothing spline estimate of median and mean absolute deviation (MAD).

Usage
```r
robustscale(y, x, nslice=400, corcenter=TRUE, decrease=TRUE)
```

Arguments
- `y` response
- `x` predictor
- `nslice` number of slices (should be "large")
- `corcenter` correct for center
- `decrease` force MAD to decrease with `x`

Details
This divides data into roughly many `nslice` slices and computes median and mean absolute deviation (mad) for each slice. These are then smoothed using `smooth.spline`.

Value
Data frame containing significant genes with the following information:
- `center` estimate of center median
- `scale` MAD estimate of scale
- `x` ordered x values for plotting
- `y` y sorted by `x`
Author(s)
Yi Lin

See Also
mad, smooth.spline

Examples

## Not run:
robustscale(y,x)

## End(Not run)

---

Simulation.pickgene  Yi Lin’s simulations for microarray analysis

Description
Example simulations

See Also
multipickgene

Examples

### Note: This uses old pickgene
#detail of the model (7-8). (first run does not include meas error \eta_i)
#par(mfrow=c(3,3))
t<-rnorm(10000,4,2)
changes1<-rep(0,10000)
changes1[1:500]<-rnorm(500)
t1<-t+changes1
changes2<-rep(0,10000)
changes2[1:500]<-rnorm(500)
t2<-t+changes2
s<-rnorm(10000,0,0.1)
cx<-3
cy<-2
t1<-t1+rnorm(10000,0,0.1)
t2<-t2+rnorm(10000,0,0.1)
x<-cx*exp(t1)
y<-cy*exp(t2)
#x<-cx*exp(t1)+rnorm(10000,0,50)
#y<-cy*exp(t2)+rnorm(10000,0,40)
xx<-qnorm(rank(x)/(10000+1))
yy<-qnorm(rank(y)/(10000+1))
#hist(x,breaks=100)
#hist(y,breaks=100)
#plot(x,y)
#hist(y[x<=0],breaks=20)
#hist(x[y<=0],breaks=20)
#plot(xx,yy)
Simulation.pickgene

topgenepick<-multipickgene( cbind(xx,yy),condi=0:1,geneID=1:10000, d=1, npickgene=500)$pick[[1]]$probe
abchangesrank<-rank((-1)*abs(t1-t2))
count <- rep(NA,500)
for( i in 1:500 ) {
    topipick <- topgenepick[i:i]
    count[i] <- sum( abchangesrank[topipick] <= i )
}

## Figure 2
plot( 1:500, 1:500, type="n",
    xlab="Rank of 500 most changed genes by our procedure",
    ylab="Number similarly ranked by the 'optimal' procedure",
    xaxs="i", yaxs="i" )
lines( 1:500, count, type="s", lty=1, lwd=2 )
abline(0,1)
## Not run: dev.print( hor=F, height=6.5, width=6.5, file="rank1.ps" )

#again, but with the additive noise. (includes \eta_i)
par(mfrow=c(2,2))
t<-rnorm(10000,4,2)
changes1<-rep(0,10000)
changes1[1:500]<-rnorm(500)
t1<-t+changes1
changes2<-rep(0,10000)
changes2[1:500]<-rnorm(500)
t2<-t+changes2
s<-rnorm(10000,0,0.1)
cx<-3
cy<-2
t1<-t1+rnorm(10000,0,0.1)
t2<-t2+rnorm(10000,0,0.1)
### note that noise is very large here (50,40)
x<-cx*exp(t1)+rnorm(10000,0,50)
y<-cy*exp(t2)+rnorm(10000,0,40)
xx<-qnorm(rank(x)/(10000+1))
yy<-qnorm(rank(y)/(10000+1))
hist(x,breaks=100)
hist(y,breaks=100)
plot(x,y,cex=0.4)
#hist(y[x<=0],breaks=20)
#hist(x[y<=0],breaks=20)
plot(xx,yy,cex=0.4)
## Not run: dev.print( hor=F, height=6.5, width=6.5, file="simudata.ps" )

topgenepick<-multipickgene(cbind(xx,yy),condi=0:1,geneID=1:10000, d=1, npickgene=500)$pick[[1]]$probe
abchangesrank<-rank((-1)*abs(t1-t2))
count <- rep(NA,500)
for( i in 1:500 ) {
    topipick <- topgenepick[i:i]
    count[i] <- sum( abchangesrank[topipick] <= i )
}
par(mfrow=c(1,1)) # figure 4
plot( 1:500, 1:500, type="n",
    xlab="Rank of 500 most changed genes by our procedure",
    ylab="Number similarly ranked by the 'optimal' procedure",
    xaxs="i", yaxs="i" )
lines( 1:500, count, type="s", lty=1, lwd=2 )
abline(0,1)
xaxs="i", yaxs="i" )
lines( 1:500, count, type="s", lty=1, lwd=2 )
abline(0,1)
## Not run: dev.print( hor=F, height=6.5, width=6.5, file="rank2.ps" )

### Figure 5

genepick <- multipickgene( cbind(xx,yy), condi=0:1, geneID=1:10000, d=1)
## Not run: dev.print( hor=F, height=6.5, width=6.5, file="simutest.ps" )$pick[[1]]$probe
npick <- length(genepick$pickedgene)
genepick$pickedgene
npick
count[npick]
Index

*Topic hplot
  oddssplot, 4
  pickgene, 5

*Topic models
  em.ggb, 1
  oddssplot, 4
  pickgene, 5

*Topic robust
  robustscale, 7

*Topic smooth
  robustscale, 7

*Topic utilities
  model.pickgene, 3

em.ggb, 1, 5
mad, 8
model.matrix, 3
model.pickgene, 3
oddssplot, 2, 4
pickgene, 5, 7
robustscale, 7
Simulation.pickgene, 8
smooth.spline, 8