Package 'MMDvariance'

October 12, 2022

Type Package

Title Detecting Differentially Variable Genes Using the Mixture of
Marginal Distributions

Version 0.0.9

Date 2018-07-22

Maintainer Weiliang Qiu <weiliang.qiu@gmail.com>

Depends R (>= 3.4.0), Biobase, lawstat

Imports MASS, graphics, stats

Suggests ALL

biocViews Bioinformatics, DifferentialExpression

Description Gene selection based on variance using the marginal distributions of gene profiles that characterized by a mixture of three-component multivariate distributions. Please see the reference: Li X, Fu Y, Wang X, DeMeo DL, Tantisira K, Weiss ST, Qiu W. (2018) <doi:10.1155/2018/6591634>.

License GPL (>= 2)

NeedsCompilation no

Author Xuan Li [aut, ctb],
Yuejiao Fu [aut, ctb],
Xiaogang Wang [aut, ctb],
Dawn L. DeMeo [aut, ctb],
Kelan Tantisira [aut, ctb],
Scott T. Weiss [aut, ctb],
Weiliang Qiu [aut, cre]

Repository CRAN

Date/Publication 2018-07-27 21:10:14 UTC

R topics documented:

gsMMD.v — *Gene selection based on variances by using a mixture of marginal distributions*

Description

Gene selection based on variances by using the marginal distributions of gene profiles that characterized by a mixture of three-component multivariate distributions. The goal is to detect gene probes having different variances between cases and controls. Input is an object derived from the class ExpressionSet. The function will obtain initial gene cluster membership by its own.

Usage

```
gsMMD.v(obj.eSet,
      memSubjects,
      maxFlag = TRUE,
      thrshPostProb = 0.5,
      geneNames = NULL,
      alpha = 0.05,
      iniGeneMethod = "myLeveneTest",
      transformFlag = FALSE,
      transformMethod = "boxcox",
      scaleFlag = TRUE,
      criterion = c("cor", "skewness", "kurtosis"),
      minL = -10,
      maxL = 10,
      stepL = 0.1,
      eps = 0.001,
      ITMAX = 100,
      plotFlag = FALSE,
      quiet=TRUE)
```

Arguments

obj.eSet — an object derived from the class ExpressionSet which contains the matrix of gene expression levels. The rows of the matrix are genes. The columns of the matrix are subjects.

memSubjects — a vector of membership of subjects. memSubjects[i]=1 means the i-th subject belongs to diseased group, 0 otherwise.

maxFlag — logical. Indicate how to assign gene class membership. maxFlag=TRUE means that a gene will be assigned to a class in which the posterior probability of the gene belongs to this class is maximum. maxFlag=FALSE means that a gene will be assigned to class 1 if the posterior probability of the gene belongs to class 1 is greater than thrshPostProb. Similarly, a gene will be assigned to class 1 if the posterior probability of the gene belongs to class 1 is greater than thrshPostProb. If the posterior probability is less than thrshPostProb, the gene will be assigned to class 2 (non-differentially variable gene group).

`thrshPostProb` threshold for posterior probabilities. For example, if the posterior probability that a gene belongs to cluster 1 given its gene expression levels is larger than `thrshPostProb`, then this gene will be assigned to cluster 1.

`geneNames` an optional character vector of gene names

`alpha` significant level which is equal to `1-conf.level`, `conf.level` is the argument for the function `t.test`.

`iniGeneMethod` method to get initial 3-cluster partition of genes: (1) genes having higher variance in cases than in controls; (2) genes having equal variance between cases and controls; (3) genes having lower variance in cases than in controls.

Available methods are: “myAWvar”, “myBFTest”, “myFTest”, “myLeveneTest”, “myLevene.TM”, “myiAWvar.BF”, “myiAWvar.Levene”, “myiAWvar.TM”, “myLeveneTest”, “myLeveneTest.TM”.

`transformFlag` logical. Indicate if data transformation is needed

`transformMethod`

method for transforming data. Available methods include "boxcox", "log2", "log10", "log", "none".

`scaleFlag` logical. Indicate if gene profiles are to be scaled to have mean zero and variance one. If `transformFlag=TRUE` and `scaleFlag=TRUE`, then scaling is performed after transformation. To avoid linear dependence of tissue samples after scaling gene profiles, we delete one tissue sample after scaling (c.f. details).

`criterion` if `transformFlag=TRUE`, `criterion` indicates what criterion to determine if data looks like normal. “cor” means using Pearson’s correlation. The idea is that the observed quantiles after transformation should be close to theoretical normal quantiles. So we can use Pearson’s correlation to check if the scatter plot of theoretical normal quantiles versus observed quantiles is a straightline. “skewness” means using skewness measure to check if the distribution of the transformed data are close to normal distribution; “kurtosis” means using kurtosis measure to check normality.

`minL` lower limit for the `lambda` parameter used in Box-Cox transformation

`maxL` upper limit for the `lambda` parameter used in Box-Cox transformation

`stepL` step increase when searching the optimal `lambda` parameter used in Box-Cox transformation

`eps` a small positive value. If the absolute value of a value is smaller than `eps`, this value is regarded as zero.

`ITMAX` maximum iteration allowed for iterations in the EM algorithm

`plotFlag` logical. Indicate if the Box-Cox normality plot should be output.

`quiet` logical. Indicate if intermediate results should be printed out.

Details

We assume that the distribution of gene expression profiles is a mixture of 3-component multivariate normal distributions $\sum_{k=1}^{3}\pi_k f_k(x|\theta)$. Each component distribution f_k corresponds to a gene cluster. The 3 components correspond to 3 gene clusters: (1) genes having higher variance in cases than in controls; (2) genes having equal variance between cases and controls; (3) genes having lower variance in cases than in controls. The model parameter vector is $\theta = (\pi_1$, π_2, π_3, σ_{c1}^2, σ_{n1}^2, μ_{c1}, ρ_{c1}, μ_{n1}, ρ_{n1}, σ_2^2, μ_{c2}, ρ_{c2}, μ_{n2}, ρ_{n2}, σ_{c3}^2, σ_{n3}^2, μ_{c3}, ρ_{c3}, μ_{n3}, ρ_{n3}. where π_1, π_2, and π_3 are the mixing proportions; μ_{c1}, σ_{c1}^2, and ρ_{c1} are the marginal mean, variance, and correlation of gene expression levels of cluster 1 (over-variable genes) for diseased subjects; μ_{n1}, σ_{n1}^2, and ρ_{n1} are the marginal mean, variance, and correlation of gene expression levels of cluster 1 (over-variable genes) for non-diseased subjects; σ_2^2, μ_{c2}, ρ_{c2}, μ_{n2}, and ρ_{n2} are the marginal mean, variance, and correlation of gene expression levels of cluster 2 (equal-variable genes); μ_{c3}, σ_{c3}^2, and ρ_{c3} are the marginal mean, variance, and correlation of gene expression levels of cluster 3 (under-variable genes) for diseased subjects; μ_{n3}, σ_{n3}^2, and ρ_{n3} are the marginal mean, variance, and correlation of gene expression levels of cluster 3 (under-variable) for non-diseased subjects.

Note that genes in cluster 2 are non-differentially variable across abnormal and normal tissue samples. Hence there are only 5 parameters for cluster 2.

To make sure the identifiability, we set the following contraints: $\sigma_{c1} > \sigma_{n1}$ and $\sigma_{c3} < \sigma_{n3}$.

To make sure the marginal covariance matrices are poisitive definite, we set the following contraints: $-1/(n_c - 1) < \rho_{c1} < 1$, $-1/(n_n - 1) < \rho_{n1} < 1$, $-1/(n-1) < \rho_2 < 1$, $-1/(n_c - 1) < \rho_{c3} < 1$, $-1/(n_n - 1) < \rho_{n3} < 1$.

We also has the following constraints for the mixing proportion: $\pi_3 = 1 - \pi_1 - \pi_2$, $\pi_k > 0$, $k = 1, 2, 3$.

We apply the EM algorithm to estimate the model parameters. We regard the cluster membership of genes as missing values.

To facilitate the estimation of the parameters, we reparametrize the parameter vector as $\theta^* = (\pi_1$, π_2, s_{c1}^2, δ_{n1}, μ_{c1}, r_{c1}, μ_{n1}, r_{n1}, s_2^2, μ_{c2}, r_{c2}, μ_{n2}, r_{n2}, s_{c3}^2, δ_{n3}, μ_{c3}, r_{c3}, μ_{n3}, $r_{n3})$, where $\sigma_{n1} = \sigma_{c1} - \exp(\delta_{n1})$, $\sigma_{n3} = \sigma_{c3} + \exp(\delta_{n3})$, $\rho_{c1} = (\exp(r_{c1}) - 1/(n_c - 1))/(1 + \exp(r_{c1}))$, $\rho_{n1} = (\exp(r_{n1}) - 1/(n_n - 1))/(1 + \exp(r_{n1}))$, $\rho_2 = (\exp(r_2) - 1/(n-1))/(1 + \exp(r_2))$, $\rho_{c3} = (\exp(r_{c3}) - 1/(n_c - 1))/(1 + \exp(r_{c3}))$, $\rho_{n3} = (\exp(r_{n3}) - 1/(n_n - 1))/(1 + \exp(r_{n3}))$.

Given a gene, the expression levels of the gene are assumed independent. However, after scaling, the scaled expression levels of the gene are no longer independent and the rank $r^* = r - 1$ of the covariance matrix for the scaled gene profile will be one less than the rank r for the un-scaled gene profile Hence the covariance matrix of the gene profile will no longer be positive-definite. To avoid this problem, we delete a tissue sample after scaling since its information has been incorrporated by other scaled tissue samples. We arbitrarily select the tissue sample, which has the biggest label number, from the tissue sample group that has larger size than the other tissue sample group. For example, if there are 6 cancer tissue samples and 10 normal tissue samples, we delete the 10-th normal tissue sample after scaling.

Value

A list contains 18 elements.

`dat` the (transformed) microarray data matrix. If tranformation performed, then `dat` will be different from the input microarray data matrix.

`memSubjects` the same as the input `memSubjects`.

`memGenes` a vector of cluster membership of genes. 1 means over-variable gene; 2 means non-differentially variable gene; 3 means under-variable gene.

`memGenes2` an variant of the vector of cluster membership of genes. 1 means differentially variable gene; 0 means non-differentially variable gene.

`para` parameter estimates (c.f. details).

`llkh` value of the loglikelihood function.

`wiMat` posterior probability that a gene belongs to a cluster given the expression levels of this gene. Column i is for cluster i.

`wiArray` posterior probability matrix for different initial gene selection methods.

`memIniMat` a matrix of initial cluster membership of genes.

`paraIniMat` a matrix of parameter estimates based on initial gene cluster membership.

`llkhIniVec` a vector of values of loglikelihood function.

`memMat` a matrix of cluster membership of genes based on the mixture of marginal models with initial parameter estimates obtained initial gene cluster membership.

`paraMat` a matrix of parameter estimates based on the mixture of marginal models with initial parameter estimates obtained initial gene cluster membership.

`llkhVec` a vector of values of loglikelihood function based on the mixture of marginal models with initial parameter estimates obtained initial gene cluster membership.

`lambda` the parameter used to do Box-Cox transformation

`paraRP` parameter estimates for reparametrized parameter vector (c.f. details).

`paraIniMatRP` a matrix of parameter estimates for reparametrized parameter vector based on initial gene cluster membership.

`paraMatRP` a matrix of parameter estimates for reparametrized parameter vector based on the mixture of marginal models with initial parameter estimates obtained initial gene cluster membership.

Note

The speed of the program is slow for large data sets.

Author(s)

Xuan Li <lixuan0759@gmail.com>, Yuejiao Fu <yuejiao@mathstat.yorku.ca>, Xiaogang Wang <stevenw@mathstat.yorku.ca>, Dawn L. DeMeo <redld@channing.harvard.edu>, Kelan Tantisira <rekgt@channing.harvard.edu>, Scott T. Weiss <restw@channing.harvard.edu>, Weiliang Qiu <weiliang.qiu@gmail.com>

References

Li X, Fu Y, Wang X, DeMeo DL, Tantisira K, Weiss ST, Qiu W. Detecting Differentially Variable MicroRNAs via Model-Based Clustering. *International Journal of Genomics*. Article ID 6591634, Volumne 2018 (2018).

Examples

```
t1 = proc.time()
library(ALL)
data(ALL)
eSet1 <- ALL[1:50, ALL$BT == "B3" | ALL$BT == "T2"]
```

```
mem.str <- as.character(eSet1$BT)
nSubjects <- length(mem.str)
memSubjects <- rep(0,nSubjects)
# B3 coded as 0, T2 coded as 1
memSubjects[mem.str == "T2"] <- 1

obj.gsMMD.v <- gsMMD.v(eSet1, memSubjects, transformFlag = FALSE,
  transformMethod = "boxcox", scaleFlag = FALSE,
  eps = 1.0e-1, ITMAX = 5, quiet = TRUE)
print(round(obj.gsMMD.v$para, 3))
t2=proc.time()-t1
print(t2)
```

`plotHistDensity.v` *Plot of histogram and density estimate of the pooled gene expression levels.*

Description

Plot of histogram of pooled gene expression levels, composited with density estimate based on the mixture of marginal distributions. The density estimate is based on the assumption that the marginal correlations between subjects are zero.

Usage

```
plotHistDensity.v(obj.gsMMD,
                  plotFlag="case",
                  plotComponent=FALSE,
                  myxlab="expression level",
                  myylab="density",
                  mytitle="Histogram (case)",
                  x.legend=NULL,
                  y.legend=NULL,
                  numPoints=500,
                  mycol=1:4,
                  mylty=1:4,
                  mylwd=rep(3,4),
                  cex.main=2,
                  cex.lab=1.5,
                  cex.axis=1.5,
                  cex=2,
                  bty="n")
```

Arguments

`obj.gsMMD` an object returned by `gsMMD.v`, `gsMMD.default.v`, `gsMMD2.v`, or `gsMMD2.default.v`

`plotFlag` logical. Indicate the plot will based on which type of subjects.

`plotComponent` logical. Indicate if components of the mixture of marginal distribution will be plotted.

`myxlab` label for x-axis

`myylab` label for y-axis

`mytitle` title of the plot

`x.legend` the x-corrdiates of the legend

`y.legend` the y-corrdiates of the legend

`numPoints` logical. Indicate how many genes will be plots.

`mycol` color for the density estimates (overall and components)

`mylty` line styles for the density estimates (overall and components)

`mylwd` line width for the density estimates (overall and components)

`cex.main` font for main title

`cex.lab` font for x- and y-axis labels

`cex.axis` font for x- and y-axis

`cex` font for texts

`bty` the type of box to be drawn around the legend. The allowed values are ’"o"’ and ’"n"’ (the default).

Details

For a given type of subjects, we pool their expression levels together if the marginal correlations among subjects are zero. We then draw a histogram of the pooled expression levels. Next, we composite density estimates of gene expression levels for the overal distribution and the 3 component distributions.

Value

A list containing coordinates of the density estimates:

`x` sorted pooled gene expression levels for cases or controls.

`x2` a subset of `x` specified by the sequence: `seq(from=1,to=len.x, by=delta)`, where `len.x` is the length of the vector `x`, and `delta=floor(len.x/numpoints)`.

`y` density estimate corresponding to `x2`

`y1` weighted density estimate for gene cluster 1

`y2` weighted density estimate for gene cluster 2

`y3` weighted density estimate for gene cluster 3

Note

The density estimate is obtained based on the assumption that the marginal correlation among subjects is zero. If the estimated marginal correlation obtained by gsMMD.v is far from zero, then do not use this plot function.

Author(s)

Xuan Li <lixuan0759@gmail.com>, Yuejiao Fu <yuejiao@mathstat.yorku.ca>, Xiaogang Wang <stevenw@mathstat.yorku.ca>, Dawn L. DeMeo <redld@channing.harvard.edu>, Kelan Tantisira <rekgt@channing.harvard.edu>, Scott T. Weiss <restw@channing.harvard.edu>, Weiliang Qiu <weiliang.qiu@gmail.com>

References

Li X, Fu Y, Wang X, DeMeo DL, Tantisira K, Weiss ST, Qiu W. Detecting Differentially Variable MicroRNAs via Model-Based Clustering. *International Journal of Genomics*. Article ID 6591634, Volumne 2018 (2018).

Examples

```
  t1 = proc.time()
  library(ALL)
  data(ALL)
  eSet1 <- ALL[1:50, ALL$BT == "B3" | ALL$BT == "T2"]

  mem.str <- as.character(eSet1$BT)
  nSubjects <- length(mem.str)
  memSubjects <- rep(0,nSubjects)
  # B3 coded as 0, T2 coded as 1
  memSubjects[mem.str == "T2"] <- 1

  obj.gsMMD.v <- gsMMD.v(eSet1, memSubjects, transformFlag = FALSE,
    transformMethod = "boxcox", scaleFlag = FALSE,
    eps = 1.0e-1, ITMAX = 5, quiet = TRUE)
  print(round(obj.gsMMD.v$para, 3))


plotHistDensity.v(obj.gsMMD.v, plotFlag = "case",
    mytitle = "Histogram (case)",
    plotComponent = TRUE,
    x.legend = c(0.8, 3),
    y.legend = c(0.3, 0.4),
    numPoints = 50)
t2=proc.time()-t1
print(t2)
```

Index