Package ‘TOAST’

October 17, 2020

Type Package

Title Tools for the analysis of heterogeneous tissues

Version 1.2.0

Description This package is devoted to analyzing high-throughput data (e.g. gene expression microarray, DNA methylation microarray, RNA-seq) from complex tissues. Current functionalities include 1. detect cell-type specific or cross-cell type differential signals 2. improve variable selection in reference-free deconvolution 3. partial reference-free deconvolution with prior knowledge.

Author Ziyi Li and Hao Wu

Maintainer Ziyi Li <ziyi.li@emory.edu>

License GPL-2

Encoding UTF-8

LazyData false

Depends R (>= 3.6), RefFreeEWAS, EpiDISH, limma, nnls

biocViews DNAMethylation, GeneExpression, DifferentialExpression, DifferentialMethylation, Microarray, GeneTarget, Epigenetics, MethylationArray

BugReports https://github.com/ziyili20/TOAST/issues

Imports stats, methods, SummarizedExperiment

Suggests BiocStyle, knitr, rmarkdown, csSAM, gplots, matrixStats, Matrix

VignetteBuilder knitr

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

git_branch RELEASE_3_11

git_last_commit b069f38

git_last_commit_date 2020-04-27

Date/Publication 2020-10-16

R topics documented:

`assignCellType` *Align cell types when reference proportions are known*

Description

Align target proportions with reference proportions by pearson correlation coefficients.

Usage

```
assignCellType(input,reference)
```

Arguments

`input`	Input proportiona matrix of dimension N by K.
`reference`	Reference proportion matrix of dimension N by K.

Value

The aligned proportion matrix, following the cell type ordering of reference proportion matrix.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhijin Wu, Peng Jin, Hao Wu. "Dissecting differential signals in high-throughput data from complex tissues."

Examples

```
## generate estimated proportion matrix
estProp <- matrix(abs(runif(50*4,0,1)), 50, 4)
estProp <- sweep(estProp, 1, rowSums(estProp), "/")

## generate reference proportion matrix
refProp <- matrix(abs(runif(50*4,0,1)), 50, 4)
refProp <- sweep(refProp, 1, rowSums(refProp), "/")

estProp_aligned = assignCellType(input = estProp,
reference = refProp)
```

CBS_PBMC_array *An example dataset for partial reference-free cell composition estimation from tissue gene expression*

Description

The dataset contains 511 microarray gene expressions for 20 PBMC samples (mixed_all) and PBMC microarray reference for the matched 511 genes from 5immune cell types (LM_5). It also contains the true cell compositions from cell sorting experiment (trueProp) and prior knowledge of cell compositions for 5 cell types in PBMC (prior_alpha and prior_sigma).

Usage

```
data("CBS_PBMC_array")
```

References

Newman, Aaron M., et al. "Robust enumeration of cell subsets from tissue expression profiles." Nature methods 12.5 (2015): 453.

Rahmani, Elior, et al. "BayesCCE: a Bayesian framework for estimating cell-type composition from DNA methylation without the need for methylation reference." Genome biology 19.1 (2018): 141.

Examples

```
data("CBS_PBMC_array")
CBS_PBMC_array$mixed_all[1:5,1:5]
head(CBS_PBMC_array$LM_5,3)
head(CBS_PBMC_array$trueProp,3)
CBS_PBMC_array$prior_alpha
CBS_PBMC_array$prior_sigma
```

ChooseMarker *Choose cell type-specific markers from pure cell type profiles or single cell data*

Description

Choose cell type-specific markers from pure cell type profiles generated by microarray or RNA-seq, or from single cell RNA-seq data by differential analysis.

Usage

```
ChooseMarker(pure_all, CellType, nMarkCT = 10, chooseSig = FALSE, verbose = TRUE)
```

Arguments

`pure_all` Input pure cell type profile matrix or single cell data matrix. Rows are for genes, columns are for cell types or cells.

`CellType` A list object consisting of cell type information for columns in pure_all. Each element is a cell type, and contains the corresponding column number in pure_all matrix. For example, CellType = list(BCell = 1:3, CD4T = 4:5).

`nMarkCT` Number of markers chosen per cell type. Default is 10.

`chooseSig` A boolean variable representing whether to consider the significance of selected markers. When chooseSig = FALSE, all nMarkerCT number of markers will be chosen per cell type. Otherwise the non-significant (p value > 0.05) markers will be filtered out.

`verbose` A boolean variable of whether to output messages.

Details

Here we provide more details for CellType variable. This variable should be a list, with each element being the corresponding column numbers in pure_all for each cell type. For example, suppose pure_all is a 1000 by 300 matrix with row being genes and column being cells (or cell types). The first 1 to 100 columns are cell A, 101 to 200 columns are cell B, and 201 to 300 columns are cell C. Then CellType should be assigned as CellType = list(A = 1:100, B = 101:200, C = 201:300). If pure_all only has three columns for three cell types A, B and C, then CellType = list(A = 1, B = 2, C = 3).

Value

A list variable, including the selected variables for all cell types.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhenxing Guo, Ying Cheng, Peng Jin, Hao Wu. "Robust partial reference-free cell compoisiton estimation from tissue expression profiles."

Examples

```
## randomly simulate pure cell type profiles
pure_all <- matrix(abs(rnorm(1000*9)), 1000, 9)
CellType <- list(CellA = 1:3,
                 CellB = 4:6,
                 CellC = 7:9)

## choose significant markers
SelMarker <- ChooseMarker(pure_all, CellType,
                          nMarkCT = 30,
                          chooseSig = TRUE,
                          verbose = FALSE)
```

csDeconv *Improve reference-free deconvolution using cross-cell type differential analysis*

Description

This function improve the feature selection in reference-free deconvolution through cross-cell type differential analysis

Usage

```
csDeconv(Y_raw, K, FUN, nMarker = 1000,
InitMarker = NULL, TotalIter = 30, bound_negative = FALSE)
```

Arguments

Y_raw A G*N matrix, G is the number of features, N is the number of subjects; or a SummarizedExperiment object.

K The number of cell types. Need to be specified a priori.

FUN The reference-free deconvolution function, this function should take Y_raw and K, and the return values should be a N by K proportion matrix. N is the number of samples and K is the number of cell types. Default function is a wrapper of the RefFreeCellMix() function from CRAN package RefFreeEWAS.

nMarker The number of markers used in the deconvolution. Default is 1000.

InitMarker A vector of length L to represent the selection of inital markers. L should be equal or smaller than G. If G is large, it is recommended that L is much smaller than G. If not specified, the most variable nMarker features will be used.

TotalIter The total number of iterations of applying cross-cell type differential analysis. Default is 30.

bound_negative Whether to bound all negative parameter estimators to zero.

Value

allProp A list of estimated proportions from all iterations.

allRMSE A vector of root mean squared errors (RMSE) from all iteratoins.

estProp A N*K matrix representing the mixture proportions of K cell types in N subjects, chosen from allProp with the smallest RMSE.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li and Hao Wu. "Improving reference-free cell composition estimation by cross-cell type differential analysis".

Examples

```
Y_raw <- abs(matrix(runif(10000*20, 0,1),10000,20))
K <- 3

## wrap your reference-free
## deconvolution method into a function
## this function should take Y and K as input
## and output a N by K proprotion matrix
## here we use RefFreeCellMix() as an example
outT <- csDeconv(Y_raw, K)

RefFreeCellMix_wrapper <- function(Y, K){
     outY = RefFreeEWAS::RefFreeCellMix(Y,
     mu0=RefFreeEWAS::RefFreeCellMixInitialize(Y,
     K = K))
     Prop0 = outY$Omega
     return(Prop0)
}

outT <- csDeconv(Y_raw, K,
            FUN = RefFreeCellMix_wrapper)
```

csTest *Testing differential signals for specified phenotype and cell type(s).*

Description

This function conducts statistical tests for specified phenotype and cell type(s).

Usage

```
csTest(fitted_model, coef = NULL, cell_type = NULL,
       contrast_matrix = NULL, var_shrinkage = TRUE,
       verbose = TRUE, sort = TRUE)
```

Arguments

`fitted_model` The output from fitModel() function.

`coef` A phenotype name, e.g. "disease", or a vector of contrast terms, e.g. c("disease", "case", "control").

`cell_type` A cell type name, e.g. "celltype1", or "neuron". If cell_type is NULL or specified as "ALL", compound effect of coef in all cell types will be tested.

`contrast_matrix`

If contrast_matrix is specified, coef and cell_type will be ignored! A matrix (or a vector) to specify contrast, e.g., cmat <- matrix(0, 2, 6); cmat[1,3] <- 1: cmat[2,4] <- 1 is to test whether the 3rd parameter and 4th parameter are zero simultaneously i.e. beta3 = beta4 = 0.

`var_shrinkage` Whether to apply shrinkage on estimated MSE or not. Applying shrinkage helps remove extremely small variance estimation and stablize statistics.

`verbose` A boolean parameter. Testing information will be printed if verbose = TRUE.

`sort` A boolean parameter. The output results will be sorted by p value if sort = TRUE.

Value

A matrix including the results from testing the phenotype in specified cell type(s).

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhijin Wu, Peng Jin, Hao Wu. "Dissecting differential signals in high-throughput data from complex tissues."

Examples

```
N <- 300 # simulation a dataset with 300 samples
K <- 3 # 3 cell types
P <- 500 # 500 features

### simulate proportion matrix
Prop <- matrix(runif(N*K, 10,60), ncol=K)
Prop <- sweep(Prop, 1, rowSums(Prop), FUN="/")
colnames(Prop) <- c("Neuron", "Astrocyte", "Microglia")

### simulate phenotype names
design <- data.frame(disease=factor(sample(0:1,
                             size = N,replace=TRUE)),
                     age=round(runif(N, 30,50)),
                     race=factor(sample(1:3, size = N,replace=TRUE)))
Y <- matrix(rnorm(N*P, N, P), ncol = N)

### generate design matrix and fit model
Design_out <- makeDesign(design, Prop)
fitted_model <- fitModel(Design_out, Y)

### check the names of cell types and phenotypes
fitted_model$all_cell_types
fitted_model$all_coefs

### detect age effect in neuron
test <- csTest(fitted_model, coef = "age",
cell_type = "Neuron", contrast_matrix = NULL)

## coef can be specified in different ways:
#### jointly test a phenotype:
test <- csTest(fitted_model, coef = "age",
cell_type = "joint", contrast_matrix = NULL)

#### if I do not specify cell_type
test <- csTest(fitted_model, coef = "age",
cell_type = NULL, contrast_matrix = NULL)
## this is exactly the same as
test <- csTest(fitted_model, coef = "age",
contrast_matrix = NULL)

#### other examples
test <- csTest(fitted_model, coef = "race",
cell_type = "Astrocyte", contrast_matrix = NULL)
test <- csTest(fitted_model, coef = "age",
cell_type = "Microglia", contrast_matrix = NULL)

#### specify contrast levels
test <- csTest(fitted_model, coef = c("race", 3, 2),
cell_type = "Neuron", contrast_matrix = NULL)
#### specify contrast levels in all cell types
test <- csTest(fitted_model, coef = c("race", 3, 2),
cell_type = "joint", contrast_matrix = NULL)

#### csTest can tolerate different ways of specifying contrast level
#### note race=1 is used as reference when fitting model
#### we can here specify race=2 as reference
test <- csTest(fitted_model, coef = c("race", 1, 2),
cell_type = "Neuron", contrast_matrix = NULL)
## get exactly the same results as
test <- csTest(fitted_model, coef = c("race", 2, 1),
cell_type = "Neuron", contrast_matrix = NULL)

#### specify a contrast matrix:
cmatrix = rep(0,15)
cmatrix[c(4,5)] = c(1,-1)
test <- csTest(fitted_model, coef = NULL,
cell_type = NULL, contrast_matrix = cmatrix)
#### specific a contrast matrix with two rows:
cmatrix = matrix(rep(0,30),2,15)
cmatrix[1,4] = 1
cmatrix[2,5] = 1
test <- csTest(fitted_model, coef = NULL,
contrast_matrix = cmatrix)
```

DEVarSelect *Feature selection for reference-free deconvolution using cross-cell type differential analysis*

Description

This function selects cross-cell type differential features for reference-free deconvolution.

Usage

```
DEVarSelect(Y_raw, Prop0, nMarker, bound_negative)
```

Arguments

`Y_raw`	A data matrix containing P features and N samples; or a SummarizedExperiment object.
`Prop0`	A N by K proportion matrix with K as number of cell types.
`nMarker`	Number of markers selected.
`bound_negative`	Whether to bound all negative parameter estimators to zero.

Value

Selected markers using cross-cell type differential analysis.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhijin Wu, Peng Jin, Hao Wu. "Dissecting differential signals in high-throughput data from complex tissues."

Examples

```
Y_raw <- matrix(runif(5000*20, 0, 1), 5000, 20)
tmp <- matrix(runif(20*4), 20, 4)
Prop0 <- sweep(tmp, 1, rowSums(tmp), "/")
varlist <- DEVarSelect(Y_raw, Prop0,
                       nMarker=1000,
                       bound_negative=FALSE)
```

findRefinx *findRefinx*

Description

Find index for marker genes with largest coefficient of variation based on raw data.

Usage

```
findRefinx(rawdata, nmarker=1000, sortBy = "var")
```

Arguments

rawdata	A data matrix with rows representing features and columns represeting samples; or a SummarizedExperiment object.
nmarker	Desired number of markers after selection. Default is 1000.
sortBy	Desired method to select features. "var" represents selecting by largest variance. "cv" represents selecting by largest coefficients of variation. Default is "var".

Value

A vector of index for the selected markers.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhijin Wu, Peng Jin, Hao Wu. "Dissecting differential signals in high-throughput data from complex tissues."

Examples

```
Y_raw <- matrix(runif(5000*20, 0, 1), 5000, 20)
idx <- findRefinx(Y_raw, nmarker=500)
idx2 <- findRefinx(Y_raw, nmarker=500, sortBy = "cv")
```

fitModel *Fit model with proportions and phenotypes.*

Description

This function receives design matrix from makeDesign() and fits the model including all cell types and phenotypes.

Usage

```
fitModel(Design_out, Y)
```

Arguments

`Design_out` The output from function makeDesign().

`Y` A G*N matrix, G is the number of features, N is the number of subjects; or a SummarizedExperiment object.

Value

`Design_out` The input Design_out object.

`N` Number of samples from matrix Y.

`coefs` Estimated coefficients (beta) in the model.

`coefs_var` Estimated variance of the coefficients (beta variance) in the model.

`Y` Observation Y matrix.

`Ypred` Predicted Y from the fitted model.

`all_coefs` The names of all phenotypes.

`all_cell_types` The names of all cell types.

`MSE` Estimated mean squared error.

`model_names` The names of all terms in the fitted model.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhijin Wu, Peng Jin, Hao Wu. "Dissecting differential signals in high-throughput data from complex tissues."

Examples

```
N = 300 # simulation a dataset with 300 samples
K = 3 # 3 cell types
P <- 500 # 500 features

### simulate proportion matrix
Prop = matrix(runif(N*K, 10,60), ncol=K)
Prop = sweep(Prop, 1, rowSums(Prop), FUN="/")
colnames(Prop) = c("Neuron", "Astrocyte", "Microglia")
Y <- matrix(rnorm(N*P, N, P), ncol = N)

### simulate phenotype names
design <- data.frame(disease=factor(sample(0:1,
                     size = N,replace=TRUE)),
                     age=round(runif(N, 30,50)),
                     race=factor(sample(1:3, size = N,replace=TRUE)))
Design_out <- makeDesign(design, Prop)

### fit model
fitted_model <- fitModel(Design_out, Y)
```

GetPrior *Get prior knowledge for supported tissue types*

Description

Users can use this function to get priors provided in this package. Users can also directly specify tisuse type in MDeconv function.

Usage

```
GetPrior(alpha = NULL, sigma = NULL)
```

Arguments

`alpha`	A string chosen from "human pbmc","human liver", "human brain", "human pancreas", "human skin", or a nuemric vector for manually specified alpha.
`sigma`	Keep it as NULL for supported tissues. Otherwise a numeric vector for manuallly specified sigma.

Value

`alpha_prior`	Prior knowledge for the mean of proportions.
`sigma_prior`	Prior knowledge for the sigma of proprotions.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhenxing Guo, Ying Cheng, Peng Jin, Hao Wu. "Robust partial reference-free cell compoisiton estimation from tissue expression profiles."

Examples

```
GetPrior("human pbmc")
GetPrior("human liver")
GetPrior("human brain")
GetPrior("human skin")
GetPrior("human pancreas")
```

makeDesign *Generate design matrix from input phenotypes and proportions.*

Description

This function generate design matrix and make preparations for following fitModel and csTest.

Usage

```
makeDesign(design, Prop)
```

Arguments

`design`	A N by P phenotype matrix, with rows as samples and columns as phenotypes (e.g. age, gender, disease, etc.).
`Prop`	A N by K proportion matrix, with rows as samples and columns as cell types

Value

`design_matrix`	A comprehensive design matrix incorporated phenotype and proportion information.
`Prop`	The input proportion matrix.
`design`	The input design/phenotype matrix.
`all_coefs`	The names of all phenotypes.
`all_cell_types`	The names of all cell types.
`formula`	The formula of the tested model, including all phenotypes, cell types and interaction terms.

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhijin Wu, Peng Jin, Hao Wu. "Dissecting differential signals in high-throughput data from complex tissues."

Examples

```
N = 300 # simulation a dataset with 300 samples
K = 3 # 3 cell types

### simulate proportion matrix
Prop = matrix(runif(N*K, 10,60), ncol=K)
Prop = sweep(Prop, 1, rowSums(Prop), FUN="/")
colnames(Prop) = c("Neuron", "Astrocyte", "Microglia")

### simulate phenotype names
design <- data.frame(disease=factor(sample(0:1, size = N,replace=TRUE)),
                     age=round(runif(N, 30,50)),
                     race=factor(sample(1:3, size = N,replace=TRUE)))
Design_out <- makeDesign(design, Prop)
```

MDeconv — *Estimate cell compoisitons via partial reference-free deconvolution.*

Description

This function is the wrapper for TOAST/-P (partial reference-free deconvolution without prior) and TOAST/+P (with prior). It guides cell compoisition estimation through extra biological information, including cell type specific markers and prior knowledge of compoisitons.

Usage

```
MDeconv(Ymat, SelMarker,
alpha = NULL, sigma = NULL,
epsilon = 0.001, maxIter = 1000,
verbose = TRUE)
```

Arguments

`Ymat` A gene expression data matrix from complex tissues. Rows represent genes and columns represent samples. Row names (e.g. gene symbol or ID) are needed. There is no need to filter data matrix by marker genes.

`SelMarker` A list variable with each element includes cell type-specific markers for this cell type. The marker list can be selected from pure cell type profiles or single cell data using ChooseMarker(). It can also be easily created manually. Please see details section below and example for more information.

`alpha` A vector including the prior mean for all cell types.

`sigma` A vector including the prior standard deviation for all cell types.

`epsilon` A numeric variable to control the level of convergence. With a large epsilon, it converges quickly and the algorithm may not converge well. With a small epsilon, it converges slower and the algorithm may converge "too much". The default value is 1e-3, which we find is a reasonable threshold.

`maxIter` Number of maximum iterations.

`verbose` A boolean variable of whether to output messages.

Details

More about SelMarker: in addition to selecting markers using ChooseMarker(), we can manually specific marker list. For example, if we know there are two cell type-specific markers for cell type A "Gene1" and "Gene2", and two cell type-specific markers for cell type B "Gene3" and "Gene4", we can create marker list by SelMarker = list(CellA = c("Gene1","Gene2"), CellB = c("Gene3","Gene4")).

One thing to note is that, the genes in marker list should have matches in row names of Ymat. If not, the unmatched markers will be removed duing analysis.

Value

A list including

H	Estimated proportion matrix, rows for cell types and columns for samples.
W	Estimated pure cell type profile matrix, rows for genes and columns for cell types

Author(s)

Ziyi Li <ziyi.li@emory.edu>

References

Ziyi Li, Zhenxing Guo, Ying Cheng, Peng Jin, Hao Wu. "Robust partial reference-free cell compoisiton estimation from tissue expression profiles."

Examples

```
# simulate mixed data from complex tissue
# without prior
Wmat <- matrix(abs(rnorm(60*3, 4, 4)), 60, 3)
SelMarker <- list(CellA = 1:20,
                  CellB = 21:40,
                  CellC = 41:60)
for(i in 1:3) {
     Wmat[SelMarker[[i]], i] <- abs(rnorm(20, 50, 10))
}
Hmat <- matrix(runif(3*25), 3, 25)
Hmat <- sweep(Hmat, 2, colSums(Hmat), "/")
Ymat <- Wmat %*% Hmat + abs(rnorm(60*25))
rownames(Ymat) <- 1:60

# deconvolution with TOAST/-P (TOAST without prior)
res <- MDeconv(Ymat, SelMarker, verbose = FALSE)
print(dim(Ymat))
cor(t(res$H), t(Hmat))

# supporse we observe the proportions
# for the same tissue from another study
alpha_prior <- rep(0.33, 3)
sigma_prior <- rep(1, 3)

# deconvolution with TOAST/+P (TOAST with prior)
res2 <- MDeconv(Ymat, SelMarker,
                alpha = alpha_prior, sigma = sigma_prior,
                verbose = FALSE)
cor(t(res2$H), t(Hmat))
```

`RA_100samples` *An example dataset for cellular proportion estimation and multiple factor design*

Description

The dataset contains normalized beta valutes for 3000 CpGs from 100 samples (50 Rheumatoid arthritis patients and 50 controls) and their phenotypes (disease status, age, and gender). The dataset also contains a sub-setted blood reference matrix for the matched 3000 CpGs. This data was obtained and processed based on GSE42861.

Usage

```
data("RA_100samples")
```

References

Liu Y, Aryee MJ, Padyukov L, Fallin MD et al. Epigenome-wide association data implicate DNA methylation as an intermediary of genetic risk in rheumatoid arthritis. Nat Biotechnol 2013 Feb;31(2):142-7. PMID: 23334450

Examples

```
data(RA_100samples)
RA_100samples$Y_raw[1:5,1:5]
head(RA_100samples$Pheno)
head(RA_100samples$Blood_ref)
```

Index